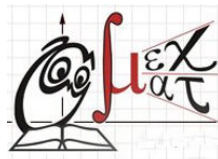




САРАТОВСКИЙ ГОСУНИВЕРСИТЕТ



МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Проектирование информационных систем

Составил: Блинков Ю. А.

Оглавление

Введение

Практическое задание

- 1 Оценка
- 2 Пример решения задачи «Телефонный справочник»
 - 2.1 Описание предметной области
 - 2.2 Первоначальная постановка задачи

Объектная модель и её реализация

CSV

База данных

XML

Web

ODF

- 2.3 Развитие постановки задачи

Объектная модель и её реализация

- 3 Набор заданий
 - 3.1 Страховая компания

- 3.2 Гостиница
- 3.3 Ломбард
- 3.4 Реализация готовой продукции
- 3.5 Ведение заказов
- 3.6 Бюро по трудоустройству
- 3.7 Нотариальная контора
- 3.8 Курсы по повышению квалификации
- 3.9 Определение факультативов для студентов
- 3.10 Распределение учебной нагрузки
- 3.11 Распределение дополнительных обязанностей
- 3.12 Техническое обслуживание станков
- 3.13 Туристическая фирма
- 3.14 Грузовые перевозки
- 3.15 Учет телефонных переговоров
- 3.16 Учет внутриофисных расходов
- 3.17 Библиотека
- 3.18 Прокат автомобилей
- 3.19 Выдача банком кредитов
- 3.20 Инвестирование свободных средств
- 3.21 Занятость актеров театра
- 3.22 Платная поликлиника
- 3.23 Анализ динамики показателей финансовой отчетности различных предприятий
- 3.24 Учет телекомпанией стоимости прошедшей в эфире рекламы
- 3.25 Интернет-магазин
- 3.26 Ювелирная мастерская

- 3.27 Парикмахерская
- 3.28 Химчистка
- 3.29 Сдача в аренду торговых площадей

1 Основные понятия технологии проектирования информационных систем

Контрольные вопросы

2 Жизненный цикл программного обеспечения ИС

Контрольные вопросы

3 Организация разработки ИС

- 3.1 Каноническое проектирование ИС
 - 3.1.1 Состав и содержание технического задания (ГОСТ 34.602-89)
 - 3.1.2 Содержание технического проекта (ГОСТ 34.602-89)
- 3.2 Типовое проектирование ИС
- 3.3 ISO/IEC 12207
- 3.4 Экстремальное программирование
 - 3.4.1 Теория
 - 3.4.2 Правила

Контрольные вопросы

4 Анализ и моделирование функциональной области внедрения ИС

- 4.1 Полная бизнес-модель компании
- 4.2 Шаблоны организационного бизнес-моделирования
- 4.3 Построения организационно-функциональной модели компании

Контрольные вопросы

5 Спецификация функциональных требований к ИС

- 5.1 Процессные потоковые модели
- 5.2 Основные элементы процессного подхода
- 5.3 Выделение и классификация процессов
- 5.4 Референтная модель бизнес-процесса
- 5.5 Проведение предпроектного обследования предприятий
- 5.6 Результаты предпроектного обследования

Контрольные вопросы

6 Методологии моделирования предметной области

- 6.1 Структурная модель
- 6.2 Функционально-ориентированные и объектно-ориентированные методологии
- 6.3 Синтетическая методика

Контрольные вопросы

7 Информационное обеспечение ИС

- 7.1 Внешнее информационное обеспечение
- 7.2 Внутримашинное информационное обеспечение

Контрольные вопросы

8 Моделирование информационного обеспечения

- 8.1 Моделирование данных
- 8.2 Создание логической модели данных
- 8.3 Проектирование хранилищ данных

Контрольные вопросы

9 Унифицированный язык визуального моделирования (UML)

9.1 Синтаксис и семантика основных объектов UML

Контрольные вопросы

10 Этапы проектирования ИС с применением UML

10.1 Разработка модели бизнес-прецедентов

10.2 Разработка модели бизнес-объектов

10.3 Разработка концептуальной модели данных

10.4 Разработка требований к системе

10.5 Разработка моделей базы данных и приложений

10.6 Проектирование физической реализации системы

Контрольные вопросы

Литература

Список иллюстраций

Список таблиц

Предметный указатель

Введение

Пособие направлено на изучение современных методов и средств проектирования информационных систем. Предусматривается изучение *CASE*-средств, как программного инструмента поддержки проектирования информационных систем (ИС).

Курс предусматривает изучение: состава и структуры различных классов экономических ИС как объектов проектирования; современных технологий проектирования ИС и методик обоснования эффективности их применения; содержания стадий и этапов проектирования ИС и их особенностей при использовании различных технологий проектирования; целей и задач проведения предпроектного обследования объектов информатизации; методов моделирования информационных процессов предметной области; классификацию и общие характеристики современных *CASE*-средств. Научной основой курса являются методологии системного анализа и моделирования, позволяющие на этапе создания информационной системы решить следующие основные задачи:

- обеспечение требуемой функциональности системы и адаптивности к изменяющимся условиям ее функционирования;
- проектирование реализуемых в системе объектов данных;
- проектирование программ и средств интерфейса (экранных форм, отчетов), которые будут обеспечивать выполнение запросов к данным;

- учет конкретной среды или технологии реализации проекта, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры, параллельной обработки, распределенной обработки данных и т.п. Программой курса предусматривается изучение CASE-инструментов поддержки проектирования информационных систем. Практикум дисциплины включает в себя задания для освоения учащимися инструментальных средств разработки и анализа функциональных и информационных моделей деятельности экономических объектов (предприятий и учреждений), являющихся основой проектирования информационных систем. Курс содержит кейс-задание на проектирование ИС для проработки в процессе изучения теоретического материала.

Цель. Дисциплина имеет целью ознакомить учащихся с информационными технологиями анализа сложных систем и основанными на международных стандартах методами проектирования информационных систем, обучить студентов принципам построения функциональных и информационных моделей систем, проведению анализа полученных результатов, применению инструментальных средств поддержки проектирования экономических информационных систем.

Практическое задание содержит варианты заданий и пример решения.

Первый раздел содержит основные понятия технологии проектирования информационных систем (ИС). Предмет и метод курса "Проектирование информационных систем". Понятие экономической информационной системы. Классы ИС. Структура однопользовательской и многопользовательской, малой и корпоративной ИС, локальной и распределенной ИС, состав и назначение подсистем. Основные особенности современных проектов ИС. Этапы создания ИС: формирование требований, концептуальное проектирование, спецификация приложений, разработка моделей, интеграция и тестирование информационной системы. Методы программной инженерии в проектировании ИС.

Во втором разделе введено понятие жизненного цикла ПО ИС. Процессы жизненного цикла: основные, вспомогательные, организационные. Содержание и взаимосвязь процессов жизненного цикла ПО ИС. Модели жизненного цикла: каскадная, модель с промежуточным контролем, спиральная.

Стадии жизненного цикла ПО ИС. Регламентация процессов проектирования в отечественных и международных стандартах.

Следующий раздел посвящен каноническому проектированию ИС. Стадии и этапы процесса канонического проектирования ИС. Цели и задачи предпроектной стадии создания ИС. Модели деятельности организации ("как есть" и "как должно быть"). Состав работ на стадии технического и рабочего проектирования. Состав проектной документации. Типовое проектирование ИС. Понятие типового проекта, предпосылки типизации. Объекты типизации. Методы типового проектирования. Оценка эффективности использования типовых решений. Типовое проектное решение (ТПР). Классы и структура ТПР. Состав и содержание операций типового элементного проектирования ИС. Функциональные пакеты прикладных программ (ППП) как основа ТПР. Адаптация типовой ИС. Методы и средства прототипного проектирования ИС.

В четвертом разделе рассмотрен анализ и моделирование функциональной области внедрения ИС. Основные понятия организационного бизнес-моделирования. Миссия компании, дерево целей и стратегии их достижения. Статическое описание компании: бизнес-потенциал компании, функционал компании, зоны ответственности менеджмента. Динамическое описание компании. Процессные потоковые модели. Модели структур данных. Полная бизнес-модель компании. Шаблоны организационного бизнес-моделирования. Построение организационно-функциональной структуры компании. Этапы разработки Положения об организационно-функциональной структуре компании. Информационные технологии организационного моделирования.

В пятом разделе дана спецификация функциональных требований к ИС. Процессные потоковые модели. Процессный подход к организации деятельности организации. Связь концепции процессного подхода с концепцией матричной организации. Основные элементы процессного подхода: границы процесса, ключевые роли, дерево целей, дерево функций, дерево показателей. Выделение и классификация процессов. Основные процессы, процессы управления, процессы обеспечения. Референтные модели. Проведение предпроектного обследования организации. Анкетирование, интервьюирование,

фотография рабочего времени персонала. Результаты предпроектного обследования.

Шестой раздел посвящен методологии моделирования предметной области. Структурная модель предметной области. Объектная структура. Функциональная структура. Структура управления. Организационная структура. Функционально-ориентированные и объектно-ориентированные методологии описания предметной области. Функциональная методика *IDEF*. Функциональная методика потоков данных. Объектно-ориентированная методика. Сравнение существующих методик. Синтетическая методика.

В следующем разделе рассмотрено информационное обеспечение ИС. Внемашинное информационное обеспечение. Основные понятия классификации информации. Понятия и основные требования к системе кодирования информации. Состав и содержание операций проектирования классификаторов. Система документации. Внутримашинное информационное обеспечение. Проектирование экранных форм электронных документов. Информационная база и способы ее организации.

В восьмом разделе представлено моделирование информационного обеспечения и моделирование данных. Метод *IDEFI*. Отображение модели данных в инструментальном средстве *ERwin*. Интерфейс *ERwin*. Уровни отображения модели. Создание логической модели данных: уровни логической модели; сущности и атрибуты; связи; типы сущностей и иерархия наследования; ключи, нормализация данных; домены. Создание физической модели: уровни физической модели; таблицы; правила валидации и значение по умолчанию; индексы; триггеры и хранимые процедуры; проектирование хранилищ данных; вычисление размера БД; прямое и обратное проектирование. Генерация кода клиентской части с помощью *ERwin*. Создание отчетов. Генерация словарей.

Унифицированный язык визуального моделирования *Unified Modeling Language (UML)* рассмотрен в девятом разделе. Диаграммы в *UML*. Классы и стереотипы классов. Ассоциативные классы. Основные элементы диаграмм взаимодействия — объекты, сообщения. Диаграммы состояний: начального состояния, конечного состояния, переходы. Вложенность состояний. Диаграммы внедрения: подсистемы, компоненты, связи. Стереотипы компонент. Диаграммы размещения.

В последнем. десятом разделе, даны этапы проектирования ИС с применением *UML*. Основные типы *UML*-диаграмм, используемые в проектировании информационных систем. Взаимосвязи между диаграммами. Поддержка *UML* итеративного процесса проектирования ИС. Этапы проектирования ИС: моделирование бизнес-прецедентов, разработка модели бизнес-объектов, разработка концептуальной модели данных, разработка требований к системе, анализ требований и предварительное проектирование системы, разработка моделей базы данных и приложений, проектирование физической реализации системы.

Практическое задание


0.1 Оценка








Для получения зачета по практике (оценка '3') необходимо полностью владеть кодом примера решения задачи «Телефонный справочник» и для своей предметной области разработать и описать диаграммы

- Прецедентов.
- Классов.
- Структуры базы данных.
- Структуры **xml**.

При этом «Первоначальная постановка задачи» — зачет за 1 семестр, «Первоначальная постановка задачи» + «Развитие постановки задачи» — зачет за 2 семестр.

Для получения по практике оценки '4', в дополнении к зачету по практике, необходимо сделать реализацию задачи в объеме примера «Телефонный справочник».

- Реализация на языке **Python** разработанной структуры классов (пример *telephonedir.py* )

- Набрать тестовые данные и организовать их чтение из формата **CSV**¹ (используя для набора текстовый редактор, пример реализации функция *load* из *tdcsv.py* ) или из базы данных **Postgresql** (используя для набора **knoda**, пример реализации функция *load* из *tddb.py* ) или из **XML** (используя для набора **kxmleditor**, пример реализации функция *load* из *tdxml.py* )
- Организовать просмотр данных через Web-интерфейс (пример *index.tpl* ) и *tdweb.py* ) или экспорт данных в ODF² (пример *tdods.py* ) и *tdodt.py* )

Для получения по практике оценки '5' необходимо

- Добавить к реализованной функции *load* функцию *save* для того же формата данных.
- Реализовать функции *load* и *save* для другого формата данных **CSV** или **Postgresql** или **XML**.

Другими словами полная реализация чтения и сохранения для 2 выбранных форматов данных. Непосредственно на занятии реализовать набор запросов предоставляемых преподавателем.

¹CSV (от англ. Comma Separated Values — значения, разделённые запятыми) — это текстовый формат, предназначенный для представления табличных данных. Каждая строка файла — это одна строка таблицы. Значения отдельных колонок разделяются разделительным символом (delimiter), например, запятой (,), точкой с запятой(;), символом табуляции. Текстовые значения обрамляются символом двойные кавычки (""); если в значении встречаются кавычки — они представляются в файле в виде двух кавычек подряд.

²OpenDocument Format (ODF, сокращённое от OASIS Open Document Format for Office Application — открытый формат документов для офисных приложений) — открытый формат файлов документов для хранения и обмена редактируемыми офисными документами, в том числе текстовыми документами (такими как заметки, отчёты и книги), электронными таблицами, рисунками, базами данных, презентациями.

0.2 Пример решения задачи «Телефонный справочник»

0.2.1 Описание предметной области

Вашей задачей является создание телефонного справочника организации.

Организация имеет различные подразделения. Каждое из них может иметь собственные подотделы. Один сотрудник может иметь несколько телефонных номеров и, наоборот, один телефон могут иметь несколько сотрудников. Необходимо создать справочник для поиска по подразделениям (подотделам), сотрудникам и телефонам.

Классы объектов

Сотрудники (Фамилия, Имя, Отчество).

Подразделения (Наименование, Сотрудники, Подотделы).

Типы телефонов (Наименование).

Телефоны (Телефон, Типа телефона, Сотрудник).

Развитие постановки задачи

Нужно учесть, что один сотрудник может работать в разных подразделениях. Например сотрудники в подразделении «ответственные за пожарную безопасность» работают и в других подразделениях (по основному месту работы).

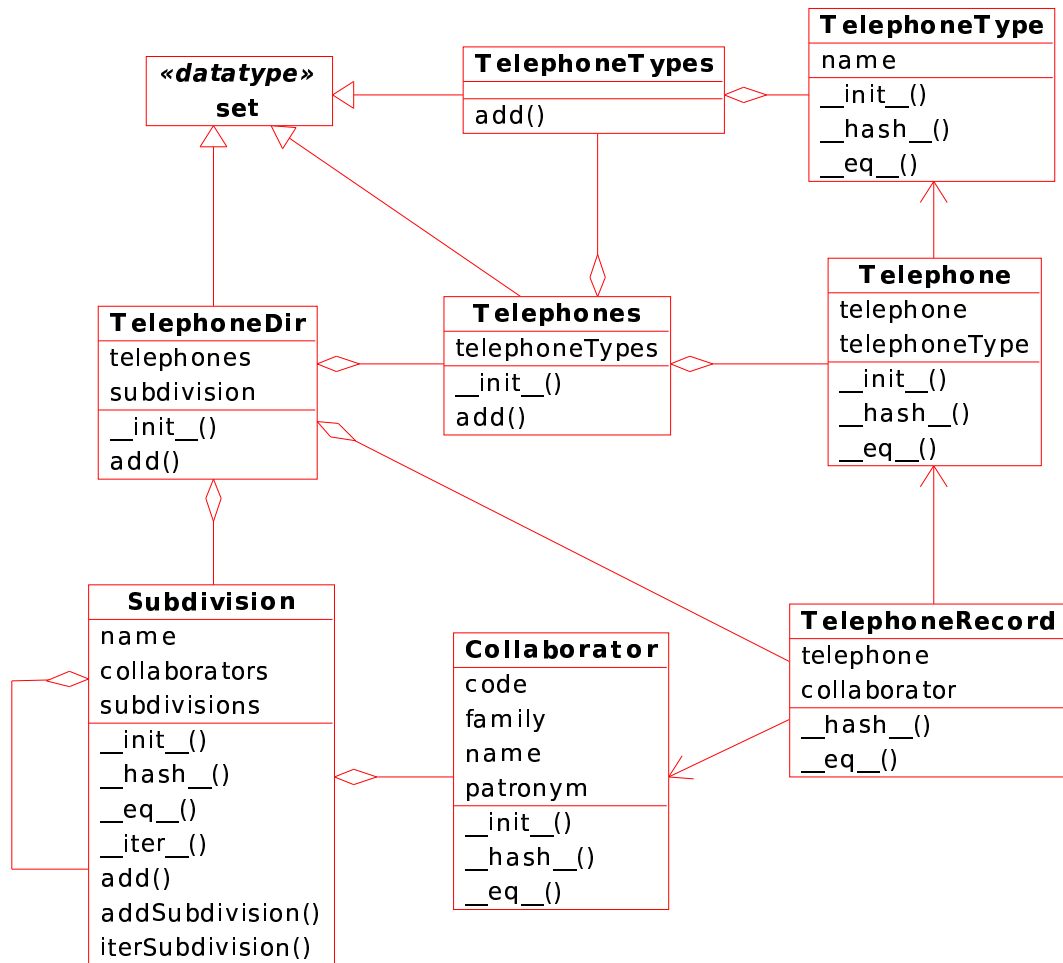
0.2.2 Первоначальная постановка задачи

Объектная модель и её реализация

Диаграмма прецедентов содержит только один прецедент “Поиск в телефонном справочнике”. Поиск должен осуществляться по подразделениям (подотделам), сотрудникам и телефонам.



Диаграмма классов значительно сложнее.



Класс *Collaborator* имеет атрибуты:

code — код сотрудника, это необходимо при одинаковых “ФИО” сотрудников или, например, при изменении фамилии сотрудника;

family — фамилия;

name — имя;

patronym — отчество.

Операции класса это:

__init__ — конструктор класса, в данном случае он необходим поскольку его атрибуты нужно инициализировать;

__hash__ — определение этой функции позволяет использовать объект класса в качестве элемента словарей и множеств;

__eq__ — поскольку хэш представляет отображение элементов некоторого множества в конечное множество чисел, обычно это числа уместяющиеся в машинное слово, то необходимо разрешать так называемые *коллизии*. В случае коллизии два разных объекта имеют одинаковое значение функции **__hash__** и **__eq__** позволяет различить эти объекты.

Подразделения представлены классом *Subdivision*, который имеет следующие атрибуты:

name — название подразделения;

collaborators — множество его сотрудников;

subdivisions — множество его собственных подразделений.

Операции класса это также состоят из конструктора, операций `__hash__` и `__eq__` поскольку подразделение может быть в качестве элемента множества *subdivisions*, а также других операций:


`__iter__` — итератор для просмотра всех сотрудников подразделения, включая и сотрудников его собственных подразделений;

add — для добавления сотрудника к подразделению;

addSubdivision — для добавления к подразделению другого подразделения в качестве его собственного подразделения;

iterSubdivision — итератор для просмотра его собственных подразделений.

Оставшиеся элементы диаграммы классов могут быть описаны кратко. *TelephoneType* определяется своим названием. *TelephoneTypes* представляет собой множество из объектов класса *TelephoneType*. Класс *Telephone* состоит из номера и его типа. Телефоны содержатся в множестве задаваемым классом *Telephones*. Телефонный справочник содержит *TelephoneRecord* ссылающийся на телефоны и подразделения.

Рассмотрим реализацию телефонного справочника . Для усвоения использованного ниже материала необходимо усвоить 1-4 лекции курса “Учебный курс - Язык программирования Python” [10] Сузи Романа Арвиевича.

```
1 #-*- coding: utf-8 -*-
2
3 """
```

```
4 Телефонная книга организации
5 """
6
7 import itertools
8
9 class Collaborator:
10     """
11     Сотрудник
12     """
13     def __init__(self, code, family, name, patronym):
14         self.code = code
15         self.family = family
16         self.name = name
17         self.patronym = patronym
18
19     def __str__(self):
20         return "%s %.2s. %.2s." % (self.family, self.name, self.patronym)
21
22     def __hash__(self):
23         return hash(self.code)
24
25     def __eq__(self, other):
26         return self.code == other.code
```

Строка 1 определяет используемую кодировку. Затем, на строках 3-5, определяем документацию нашей программы. На 7 строке импортируем модуль *itertools* для организации итераторов для обхода подразделения по его сотрудникам и по его собственным подразделениям. Далее следует описание

класса *Collaborator*. При вычисление хэш используется стандартная *hash* функция языка *Python*.

```
28 class Subdivision:
29     """
30     Подразделение
31     """
32     def __init__(self, name):
33         self.name = name
34         self.collaborators = set()
35         self.subdivisions = set()
36
37     def __hash__(self):
38         return hash(self.name)
39
40     def __eq__(self, other):
41         return self.name == other.name
42
43     def __iter__(self):
44         i = iter(self.collaborators)
45         for s in self.subdivisions:
46             i = itertools.chain(i, iter(s))
47         return i
48
49     def add(self, collaborator):
50         assert collaborator not in self
51         self.collaborators.add(collaborator)
52
```

```

53 def addSubdivision(self, subdivision):
54     assert subdivision not in self.subdivisions
55     assert not set(self).intersection(set(subdivision))
56     self.subdivisions.add(subdivision)
57
58 def iterSubdivision(self):
59     i = iter(self.subdivisions)
60     for s in self.subdivisions:
61         i = itertools.chain(i, s.iterSubdivision())
62     return i

```

В *Subdivision* для организации итераторов `__iter__` и *iterSubdivision* используется так называемое зацепление итераторов. Операция *add* на строчке 50 запрещает добавлять сотрудника к подразделению если он есть в нем или в его собственном подразделении. Это достигается за счет переопределения стандартного метода `__iter__` для контейнера языка *Python*. Строчка 54 запрещает добавлять подразделение если там оно уже есть, а строчка 55 если в нем есть хотя бы один сотрудник уже имеющиеся в нашем подразделении.

```

64 class TelephoneType:
65     """
66     Тип телефона
67     """
68     def __init__(self, name):
69         self.name = name
70
71     def __hash__(self):
72         return hash(self.name)

```

```
73
74 def __eq__(self, other):
75     return self.name == other.name
```

Тип телефона определяется его именем.

```
77 class TelephoneTypes(set):
78     """
79     Типы телефонов
80     """
81     def add(self, telephoneType):
82         assert telephoneType not in self
83         set.add(self, telephoneType)
```

На строке 82 стоит запрет на добавление к множеству типов телефонов элемента если он там уже есть.

```
85 class Telephone:
86     """
87     Телефон
88     """
89     def __init__(self, telephone, telephoneType):
90         self.number = telephone
91         self.type = telephoneType
92
93     def __hash__(self):
94         return hash(self.number)
95
```

```
96 def __eq__(self, other):
97     return self.number == other.number
```

Телефон определяется его типом и номером.

```
99 class Telephones(set):
100     """
101     Телефоны
102     """
103     def __init__(self, telephoneTypes):
104         set.__init__(self)
105         self.telephoneTypes = telephoneTypes
106
107     def add(self, telephone):
108         assert telephone not in self
109         assert telephone.type in self.telephoneTypes
110         set.add(self, telephone)
```

На строке 108 стоит запрет на добавление к множеству телефонов элемента с уже существующим номером или и на строке 109 если его типа нет в *telephoneTypes*.

```
112 class TelephoneRecord:
113     """
114     Запись в телефонном справочнике
115     """
116     def __init__(self, telephone, collaborator):
117         self.telephone = telephone
118         self.collaborator = collaborator
```

```
119
120 def __hash__(self):
121     return hash((self.telephone, self.collaborator))
122
123 def __eq__(self, other):
124     return self.telephone == other.telephone and \
125         self.collaborator == other.collaborator
```

Запись в телефонном справочнике должна ссылаться на телефон и сотрудника.

```
127 class TelephoneDir(set):
128     """
129     Телефонный справочник
130     """
131     def __init__(self, telephones, subdivision):
132         set.__init__(self)
133         self.telephones = telephones
134         self.subdivision = subdivision
135
136     def add(self, telephoneRecord):
137         assert telephoneRecord.telephone in self.telephones
138         assert telephoneRecord.collaborator in self.subdivision
139         assert telephoneRecord not in self
140         set.add(self, telephoneRecord)
```

На строке 137 стоит запрет на добавление к множеству телефона, если его нет среди телефонов справочника. Аналогично, на строке 138, запрет для подразделений. Строка 139 запрещает добавлять записи в телефонный справочник, если они уже там есть.


```

142 if __name__ == '__main__':
143     import tdcsv
144
145     telephoneDir = tdcsv.load()
146
147     for s in telephoneDir.subdivision.iterSubdivision():
148         if s.name == 'помощник проректора':
149             for r in telephoneDir:
150                 if r.collaborator in s and r.collaborator.family.find('сх') >= 0:
151                     print r.telephone.number, "%s %s. %s." % \
152                         (r.collaborator.family, r.collaborator.name[:2],
r.collaborator.patronym[:2])
153                     break
154
155     for s in telephoneDir.subdivision.iterSubdivision():
156         if s.name == 'зав. кафедрой':
157             for r in telephoneDir:
158                 if r.collaborator in s and r.collaborator.family.find('сх') >= 0:
159                     print r.telephone.number, "%s %s. %s." % \
160                         (r.collaborator.family, r.collaborator.name[:2],
r.collaborator.patronym[:2])
161                     break

```

Каждый модуль языка *Python* может быть использован в качестве головного, в этом случае значение переменной `__name__` будет иметь значение `'__main__'` (см. строку 142).

На 143 строке подключаем модуль импорта данных из формата **CSV**, а на 145 осуществляется сам импорт.

На строчках 147–153 осуществляется поиск в подразделении *помощник проректора* сотрудника, фамилия которого содержит *ск*.

На строчках 155–161 осуществляется поиск, аналогичный предыдущему, в подразделении *зав. кафедрой* сотрудника, фамилия которого содержит *сс*.

Результат работы программы представлен ниже.



51–57–39 Виноградский С. Г.

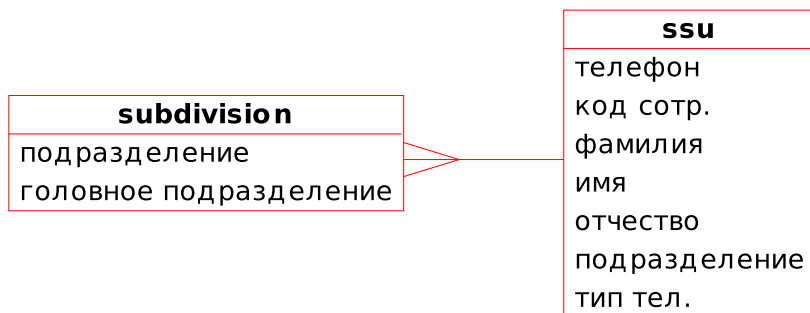
51–18–84 Лосатинская А. С.

В результате первый запрос вернул два телефона, а второй не одного.

CSV

Для усвоения материала необходимо ознакомиться с 7 курса “Учебный курс - Язык программирования Python” [10] Сузи Романа Арвиевича.

Для представления информации используется два файла *subdivision.csv*  и *ssu.csv*  со структурой колонок представленных на рисунке ниже.





Для набора можно воспользоваться любым текстовым редактором поддерживающим кодировку *UTF8* или экспортом из электронных таблиц.

Сначала опишем функцию *load* из файла *tdcsv.py* для загрузки данных из формата **CSV**.

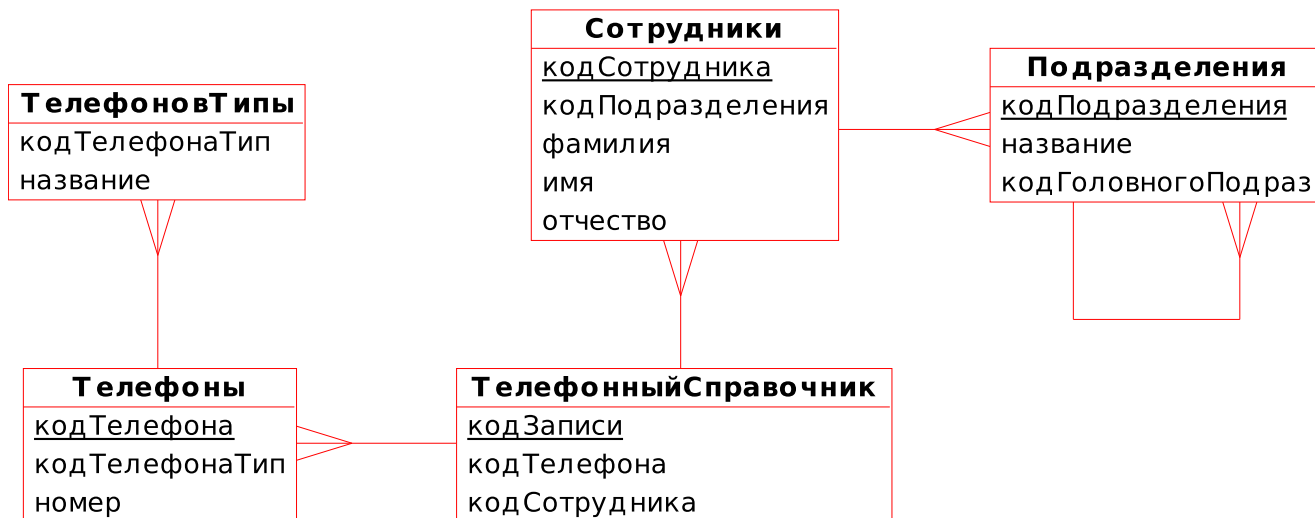
```
30 def load():
31     subdivision = {}
32     for rec in csv.reader(open(os.path.join(os.getcwd(), 'subdivision.csv'),
33 'rb'), delimiter=';'):
34         subdivision[rec[0]] = telephonedir.Subdivision(rec[0])
35         if rec[1]:
36             subdivision[rec[1]].addSubdivision(subdivision[rec[0]])
37         else:
38             telephoneDir =
39 telephonedir.TelephoneDir(telephonedir.Telephones(telephonedir.TelephoneTypes()),
40 subdivision[rec[0]])
41
42 telephones, telephoneTypes, collaborators = {}, {}, {}
43 for rec in csv.reader(open(os.path.join(os.getcwd(), 'ssu.csv'), 'rb'),
44 delimiter=';'):
45     if rec[6] not in telephoneTypes:
46         telephoneTypes[rec[6]] = telephonedir.TelephoneType(rec[6])
47         telephoneDir.telephones.telephoneTypes.add(telephoneTypes[rec[6]])
48     if rec[0] not in telephones:
49         telephones[rec[0]] = telephonedir.Telephone(rec[0],
50 telephoneTypes[rec[6]])
51         telephoneDir.telephones.add(telephones[rec[0]])
52     key = int(rec[1])
```


```
48     if key not in collaborators:
49         collaborators[key] = telephonedir.Collaborator(key, rec[2], rec[3],
rec[4])
50         subdivision[rec[5]].add(collaborators[key])
51     telephoneDir.add(telephonedir.TelephoneRecord(telephones[rec[0]],
collaborators[key]))
52
53     return telephoneDir
```

Словарь *subdivision* на строке 31 будем использовать для хранения объектов – подразделений, читаемых из файла 'subdivision.csv' . Аналогично на строке 39 организовано чтение из файла 'ssu.csv' . Окончательно функция возвращает заполненный телефонный справочник. Функция *save* сохраняет данные в эти же файлы.

База данных

Структура базы данных представлена ниже



Для создания таблиц используется следующий **SQL** код для базы данных **Postgresql** из файла *telephonedir.sql* .

```


1 CREATE TABLE Подразделения (
2   кодПодразделения SERIAL PRIMARY KEY,
3   название TEXT,
4   кодГоловногоПодраз INTEGER REFERENCES Подразделения (кодПодразделения)
5   DEFAULT NULL,
6   UNIQUE(название, кодГоловногоПодраз)
7 );

```




```

8 CREATE TABLE Сотрудники (
9     кодСотрудника INTEGER PRIMARY KEY,
10    кодПодразделения INTEGER REFERENCES Подразделения (кодПодразделения)
DEFAULT NULL,
11    фамилия TEXT,
12    имя TEXT,
13    отчество TEXT
14 );
15
16 CREATE TABLE ТелефоновТипы (
17    кодТелефонаТип SERIAL PRIMARY KEY,
18    название TEXT,
19    UNIQUE(название)
20 );
21
22 CREATE TABLE Телефоны (
23    кодТелефона SERIAL PRIMARY KEY ,
24    кодТелефонаТип INTEGER REFERENCES ТелефоновТипы(кодТелефонаТип) ,
25    номер TEXT
26 );
27
28 CREATE TABLE ТелефонныйСправочник (
29    кодЗаписи SERIAL PRIMARY KEY,
30    кодТелефона INTEGER REFERENCES Телефоны(кодТелефона) ,
31    кодСотрудника INTEGER REFERENCES Сотрудники(кодСотрудника) ,
32    UNIQUE(кодТелефона, кодСотрудника)
33 );


```

С помощью скрипта *create-db*  написанного на языке **BASH** создается база данных *telephonedir1* со структурой таблиц задаваемых представленным выше **SQL** кодом.

Набор данных можно осуществить с помощью **knoda** – графического интерфейса пользователя для доступа к базам данных.

С помощью скриптов *dump*  можно сохранить базу данных в файле *dump.tar* , а с помощью *restore*  её восстановить.

Для усвоения материала необходимо ознакомиться с 10 лекцией “Учебный курс - Язык программирования Python” [10] Сузи Романа Арвиевича.

Для работы с **Postgresql** на строке 3 файла *tddb.py*  подключим модуль *psycopg2*. На строке 7 создается соединение с базой данных, а на строке 13 закрытие этого соединения.

```
1  # -*- coding: utf-8 -*-
2
3  import psycopg2 as db
4
5  import telephonedir
6
7  def createConn():
8      conn = db.connect(host='localhost', database="telephonedir1",
user="postgres")
9      db.threadsafety=2
10     curs = conn.cursor()
11     return (db, conn, curs)
12
```

```
13 def closeConn(db, conn, curs):
14     conn.close()
```

На строке 87 выполняется запрос в базе данных. В цикле *for* на строке 88 выполняется его просмотр. Заметим, что модуль *psycopg2* поддерживает спецификацию *DBI-2.0* и поэтому кортеж *rec* содержит элементы преобразованные из базы данных согласно их типам. Так например целое из базы данных перейдет в целое языка **Python**, аналогичное преобразование будет выполнено и для чисел с плавающей точкой, строк. Значение *NULL* будет представлено в **Python** значением *None*.

```
75 def load():
76     db, conn, curs = createConn()
77
78     sql = """
79     SELECT
80         кодПодразделения,
81         название,
82         кодГоловногоПодраз
83     FROM
84         Подразделения"""
85     subdivisions, subdivisionOwners = {}, {}
86     try:
87         curs.execute(sql)
88         for rec in curs.fetchall():
89             s = telephonedir.Subdivision(rec[1])
90             subdivisions[rec[0]] = s
91             subdivisionOwners[s] = rec[2]
92     except db.DatabaseError, x:
```



```
93     print x
94     conn.rollback()
95 else:
96     conn.commit()
97 for s, i in subdivisionOwners.iteritems():
98     if i:
99         subdivisions[i].subdivisions.add(s)
100     else:
101         subdivision = s
102
103     sql = """
104 SELECT
105     кодСотрудника,
106     кодПодразделения,
107     фамилия,
108     имя,
109     отчество
110 FROM
111     Сотрудники"""
112     collaborators = {}
113     try:
114         curs.execute(sql)
115         for rec in curs.fetchall():
116             c = telephonedir.Collaborator(rec[0], rec[2], rec[3], rec[4])
117             collaborators[rec[0]] = c
118             subdivisions[rec[1]].add(c)
119     except db.DatabaseError, x:
```

```
120     print x
121     conn.rollback()
122 else:
123     conn.commit()
124
125     sql = """
126 SELECT
127     кодТелефонаТип,
128     название
129 FROM
130     ТелефоновТипы"""
131     telephoneTypes = {}
132     try:
133         curs.execute(sql)
134         for rec in curs.fetchall():
135             telephoneTypes[rec[0]] = telephonedir.TelephoneType(rec[1])
136     except db.DatabaseError, x:
137         print x
138         conn.rollback()
139     else:
140         conn.commit()
141
142     sql = """
143 SELECT
144     кодТелефона,
145     кодТелефонаТип,
146     номер
```

```

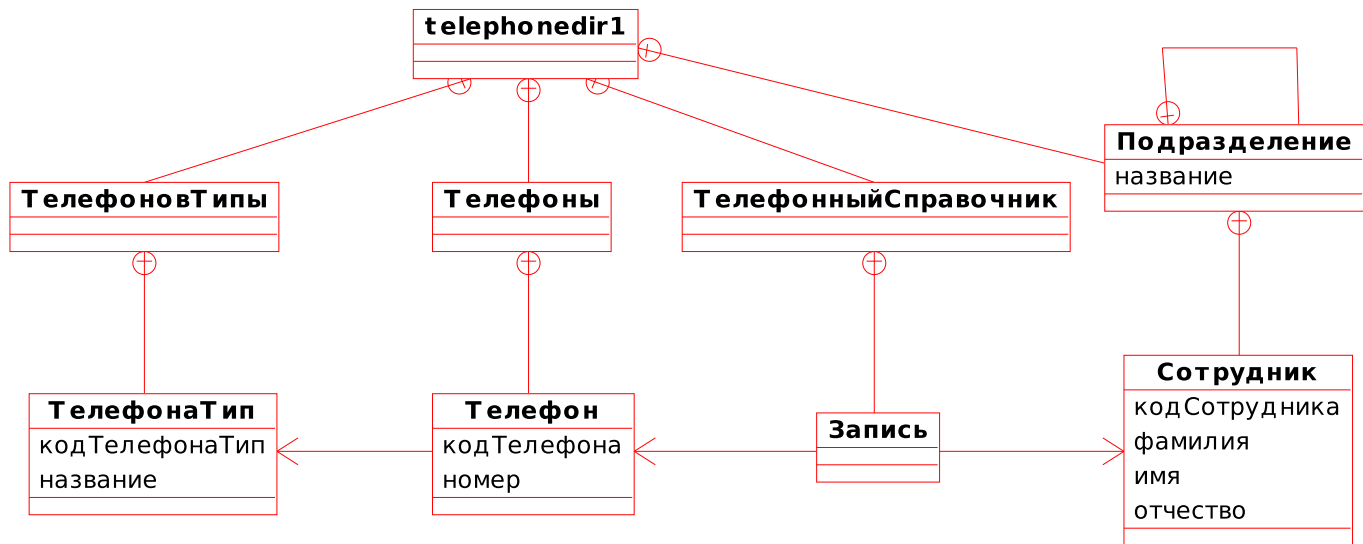
147 FROM
148     Телефоны"""
149     telephones = {}
150     try:
151         curs.execute(sql)
152         for rec in curs.fetchall():
153             telephones[rec[0]] = telephonedir.Telephone(rec[2],
telephoneTypes[rec[1]])
154     except db.DatabaseError, x:
155         print x
156         conn.rollback()
157     else:
158         conn.commit()
159
160     telephoneDir =
telephonedir.TelephoneDir(telephonedir.Telephones(telephonedir.TelephoneTypes()),
subdivision)
161     for t in telephoneTypes.itervalues():
162         telephoneDir.telephones.telephoneTypes.add(t)
163     for t in telephones.itervalues():
164         telephoneDir.telephones.add(t)
165
166     sql = """
167 SELECT
168     кодТелефона,
169     кодСотрудника
170 FROM


```

```
171     ТелефонныйСправочник"""
172     try:
173         curs.execute(sql)
174         for rec in curs.fetchall():
175             telephoneDir.add(telephonedir.TelephoneRecord(telephones[rec[0]],
collaborators[rec[1]]))
176     except db.DatabaseError, x:
177         print x
178         conn.rollback()
179     else:
180         conn.commit()
181
182     closeConn(db, conn, curs)
183
184     return telephoneDir
```


XML

Структура **XML** файла представлена ниже




Для заполнения файла *telephonedir.xml*  можно применять **kxmleditor**.

Для усвоения материала необходимо ознакомиться с 7 лекцией “Учебный курс - Язык программирования Python” [10] Сузи Романа Арвиевича.

Сузи Романа Арвиевича. При чтении из файла *telephonedir.xml* функция *load* из файла *tdxml.py*  использует технологию **SAX**, а при сохранении в **XML** функция *save* технологию **DOM**.

Web

Модуль *tdweb.py*  предоставляет web-интерфейс для быстрого поиска по подразделениям, сотрудникам и телефонам.

Телефонный справочник - Mozilla Firefox

Файл Правка Вид Журнал Закладки Инструменты Справка

http://localhost:8080/ Перейти Google

Телефонных номеров: 20

По подразделению По сотруднику По номеру По типу

ректорат все

| # | ФИО | Телефон | |
|----|--------------------|----------|-----|
| | | Номер | Тип |
| 1 | Коссович Л. Ю. | 26-16-96 | раб |
| 2 | Абрамейцева В. В. | 51-16-35 | раб |
| 3 | Сучков С. Г. | 51-72-06 | раб |
| 4 | Павлова Ю. А. | 26-16-96 | раб |
| 5 | Первушов Е. М. | 51-18-84 | раб |
| 6 | Усанов Д. А. | 27-14-96 | раб |
| 7 | Лосатинская А. С. | 51-18-84 | раб |
| 8 | Цыцина И. В. | 52-29-12 | раб |
| 9 | Портянкин В. Н. | 26-03-27 | раб |
| 10 | Митрохин В. А. | 51-18-85 | раб |
| 11 | Малинский И. Г. | 27-78-63 | раб |
| 12 | Пузикова М. А. | 51-18-85 | раб |
| 13 | Монахов С. Ю. | 51-72-06 | раб |
| 14 | Федотов Н. Ю. | 26-03-27 | раб |
| 15 | Овчинников И. С. | 51-17-41 | раб |
| 16 | Лямин М. В. | 51-36-99 | раб |
| 17 | Дергунов А. В. | 26-02-04 | раб |
| 18 | Филатова И. Н. | 26-02-41 | раб |
| 19 | Шмойлов С. С. | 51-57-39 | раб |
| 20 | Виноградский С. Г. | 51-57-39 | раб |

Предыдущая страница | Следующая страница

[вернуться на главную страницу](#)

Готово


Командная строка для запуска: `python tdweb.py`

Просмотр на своем компьютере `http://localhost:8080`.

Просмотр на удаленном компьютере `http://"ip адрес":8080`

Для выключения web-сервера необходимо нажать в консоли запуска "Ctrl-C"или перезагрузить/выключить компьютер.

Web-справочник базируется на двух дополнительных модулей языка Python: Cheetah и CherryPy.

Файл `index.tmpl`  представляет собой шаблон основной html-страницы. Это страница в коде `html` содержит передаваемые шаблону переменные начинающиеся со знака `$` или внутри конструкций `$()` или `${}`. Управляющие конструкции начинаются с `#`.

```
1 #encoding utf-8
2 <html>
3 <head><meta http-equiv='Content-Type' content='text/html; charset=utf-8'>
4 <title>Телефонный справочник</title>
5 <link rel=stylesheet type=text/css href='/style.css'>
6 </head>
7 <body>
8 <hr size=1 color=#000066>
9 <p>Телефонных номеров: <b>${len($telephoneDir)}</b></p>
10 <form action='/index' method=get>
11   <table cellspacing=8 cellpadding=0>
12     <tr>
13       <td align=center>
14         По подразделению
15       </td>
16       <td align=center>
17         По сотруднику
```



```
18     </td>
19     <td align=center>
20         По номеру
21     </td>
22     <td align=center>
23         По типу
24     </td>
25     <td align=center rowspan=2 valign=bottom>
26         <input type=submit value="Задать фильтр">
27     </td>
28 </tr>
29 <tr>
30     <td align=center>
31         <select name=subdivision class=inp>
32             #if $subdivision == '0'
33                 <option value='0' selected>все</option>
34             #else
35                 <option value='0'>все</option>
36             #end if
37             #for $i in range(1, len($subdivisions))
38                 #if $i == $subdivision
39                     <option value='$i' selected>$(subdivisions[$i].name)</option>
40                 #else
41                     <option value='$i'>$(subdivisions[$i].name)</option>
42                 #end if
43             #end for
44         </select>
```

```
45         </td>
46         <td align=center>
47             <input class=inp type=text name=collaborator value='${collaborator}'
size=12 maxlength=13>
48         </td>
49         <td align=center>
50             <input class=inp type=text name=number value='${number}' size=12
maxlength=13>
51         </td>
52         <td align=center>
53             <select name=telephoneType class=inp>
54                 #if $telephoneType == '0'
55                     <option value='0' selected>Bce</option>
56                 #else
57                     <option value='0'>Bce</option>
58                 #end if
59                 #for $i in range(1, len($telephoneTypes))
60                     #if $i == $telephoneType
61                         <option value='$i' selected>$(telephoneTypes[$i].name)</option>
62                     #else
63                         <option value='$i'>$(telephoneTypes[$i].name)</option>
64                     #end if
65                 #end for
66             </select>
67         </td>
68     </tr>
69 </table>
70 </form>
```

```


70
71 #set $pageSize = 20
72 <table border=0 cellpadding=2>
73   <tr class=rowhead align=center>
74     <td align=center rowspan=2><b>#</b></td>
75     <td align=center rowspan=2><b>ФИО</b></td>
76     <td align=center colspan=2><b>Телефон</b></td>
77   </tr>
78   <tr class=rowhead align=center>
79     <td align=center><b>Home</b></td>
80     <td align=center><b>Тип</b></td>
81   </tr>
82   #set $r=2
83   #for i in range($page*$pageSize, min(($page+1)*$pageSize,
len($telephoneDir)))
84     <tr class=row${r}>
85       <td align=right>${i+1}</td>
86       <td align=left>${telephoneDir[i].collaborator}</td>
87       <td align=left>${telephoneDir[i].telephone.number}</td>
88       <td align=center>${telephoneDir[i].telephone.type.name}</td>
89     </tr>
90     #if $r == 1
91       #set $r = 2
92     #else
93       #set $r = 1
94     #end if
95   #end for

```

```

96 </table>
97
98 #if $page == 0
99     #set $prevPage = "Предыдущая страница"
100 #else
101     #set $prevPage = "<A
href='/index?page=%d&subdivision=%d&collaborator=%s&number=%s&telephoneType=%d'>Предыд
страница</A>" % \
102         ($page - 1, $subdivision, $collaborator, $number,
$telephoneType)
103 #end if
104 #if ($page + 1)*$pageSize >= len($telephoneDir)
105     #set $nextPage = "Следующая страница"
106 #else
107     #set $nextPage = "<A
href='/index?page=%d&subdivision=%d&collaborator=%s&number=%s&telephoneType=%d'>Следую
страница</A>" % \
108         ($page + 1, $subdivision, $collaborator, $number,
$telephoneType)
109 #end if
110 <br>${prevPage}&nbsp;|&nbsp;${nextPage}
111 <p><hr size=1 color=#000066><a href="/">вернутся на главную страницу</a>
112 </body>
113 </html>

```


В модуле `tdweb.py`  на строке 21 описан декоратор `cherry.py.expose`, показывающий что следующая функция будет представлять собой страницу на сайте с аргументами соответствующими

входным переменным функции.




```
1  #-*- coding: utf-8 -*-
2
3  """
4  Web интерфейс телефонной книги
5  """
6
7  import os
8  from Cheetah.Template import Template
9  import cherrypy
10
11  class Root:
12      @staticmethod
13      def prepare(telephoneDir):
14          Root.collaborators = list(sorted(telephoneDir.subdivision))
15          Root.subdivisions =
list(sorted(telephoneDir.subdivision.iterSubdivision()))
16          Root.subdivisions.insert(0, 'все')
17          Root.telephoneTypes =
list(sorted(telephoneDir.telephones.telephoneTypes))
18          Root.telephoneTypes.insert(0, 'все')
19          Root.telephoneDir = list(sorted(telephoneDir))
20
21      @cherrypy.expose
22      def index(self, page='0', subdivision='0', collaborator='', number='',
telephoneType='0'):
```

```
23     page = int(page)
24     subdivision = int(subdivision)
25     telephoneType = int(telephoneType)
```

ODF

Экспорт данных в модуле *tdods.py*  в формате *ods*³ очень похож на сохранения данных в формате **CSV**. Для экспорта используется модуль *odf* языка *Python*, а точнее набор функций и классов описанных ниже в строках 3–6.

```
1  # -*- coding: utf-8 -*-
2
3  from odf.opendocument import OpenDocumentSpreadsheet
4  from odf.style import Style, TextProperties, ParagraphProperties,
TableColumnProperties
5  from odf.text import P
6  from odf.table import Table, TableColumn, TableRow, TableCell
```

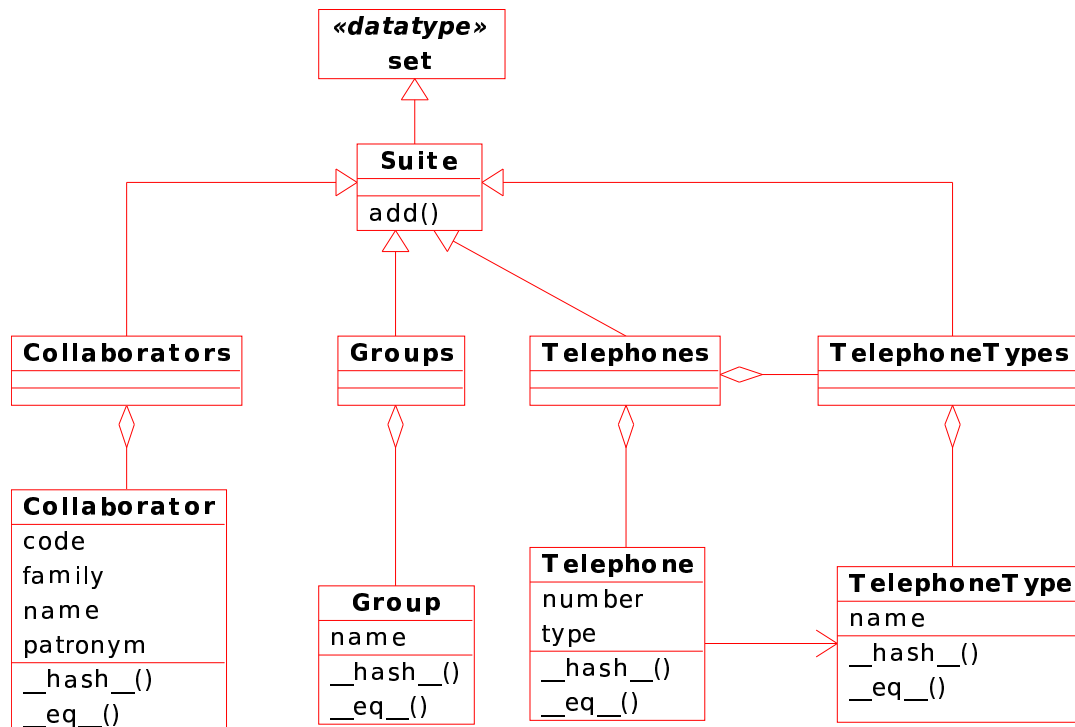
Сохранение запроса в формате *odt*⁴ в модуле *tdodt.py*  с точки зрения запроса похож на разработанный модуль *tdweb.py* , а в смысле сохранения на предыдущий *tdods.py* .

³**ODF** документ “электронная таблица” с расширением *ods*

⁴**ODF** документ “текстовый документ” с расширением *odt*

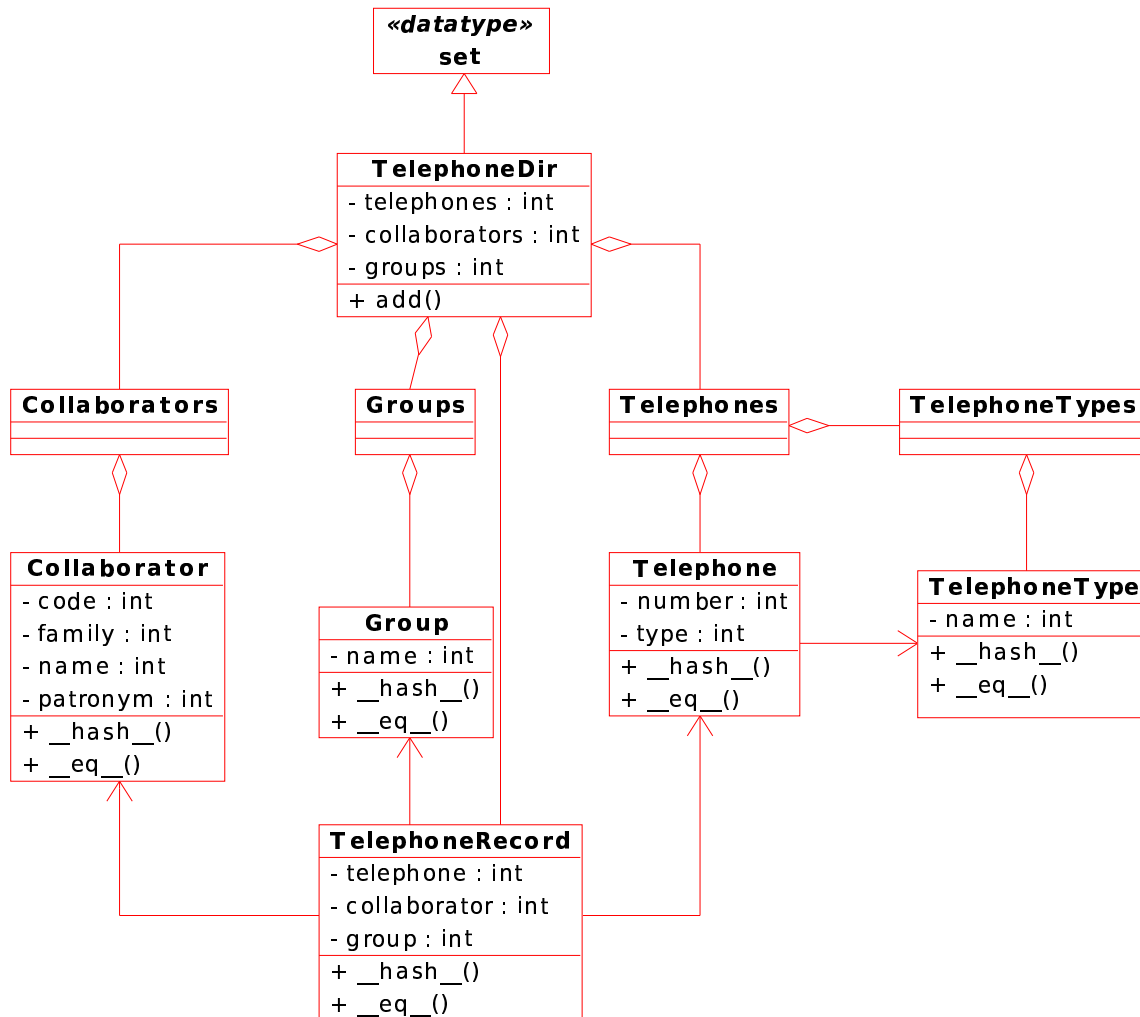
0.2.3 Развитие постановки задачи

Диаграмма прецедентов остается без изменений. Диаграмму классов лучше разбить на две.





Поскольку сотрудники могут одновременно входить в разные подразделения, заменим *Subdivision* на *Group*. Также *Subdivision* образовывали иерархическую структуру, а *Group* нет, поэтому для

организации хранения введем множество групп *Groups*. Из предыдущего решения задачи видно, что *Collaborators*, *TelephoneTypes*, *Telephones* и теперь и *Groups* обладают общими свойствами и поведением, а именно все они множества с запретом на добавление элемента если он уже там есть. Поэтому они могут быть порождены от общего класса *Suite* (Набор), который в свою очередь порожден от стандартного типа языка *Python set*. Атрибуты и операции классов уже описаны в предыдущей задаче.



Объектная модель и её реализация

Изменения, по сравнению с предыдущей задачей, коснулись в основном классов *TelephoneRecord*, *TelephoneDir*. Это связано с тем, что теперь запись  в телефонном справочнике  представляет собой набор из телефона, сотрудника и группы. Соответственно *TelephoneDir* содержит *Telephones*, *Collaborators* и *Groups*.

```
1  #-*- coding: utf-8 -*-
2
3  """
4  Телефонная книга организации
5  """
6
7  class Suite(set):
8      """
9      Множество с запретом на добавление элементов
10     уже содержащихся в нём
11     """
12     def add(self, elt):
13         assert elt not in self
14         set.add(self, elt)
15
16 class Collaborator:
17     """
18     Сотрудник
19     """
20     def __init__(self, code, family, name, patronym):
```

```
21     self.code = code
22     self.family = family
23     self.name = name
24     self.patronym = patronym
25
26     def __hash__(self):
27         return hash(self.code)
28
29     def __eq__(self, other):
30         return self.code == other.code
31
32 class Collaborators(Suite):
33     """
34     Сотрудники
35     """
36     pass
37
38 class Group:
39     """
40     Группа сотрудников
41     """
42     def __init__(self, name):
43         self.name = name
44
45     def __hash__(self):
46         return hash(self.name)
47
```

```
48     def __eq__(self, other):
49         return self.name == other.name
50
51 class Groups(Suite):
52     """
53     Группы
54     """
55     pass
56
57 class TelephoneType:
58     """
59     Тип телефона
60     """
61     def __init__(self, name):
62         self.name = name
63
64     def __hash__(self):
65         return hash(self.name)
66
67     def __eq__(self, other):
68         return self.name == other.name
69
70 class TelephoneTypes(Suite):
71     """
72     Типы телефонов
73     """
74     pass
```

```
75
76 class Telephone:
77     """
78     Телефон
79     """
80     def __init__(self, telephone, telephoneType):
81         self.number = telephone
82         self.type = telephoneType
83
84     def __hash__(self):
85         return hash(self.number)
86
87     def __eq__(self, other):
88         return self.number == other.number
89
90 class Telephones(set):
91     """
92     Телефоны
93     """
94     def __init__(self, telephoneTypes):
95         set.__init__(self)
96         self.telephoneTypes = telephoneTypes
97
98     def add(self, telephone):
99         assert telephone not in self
100         assert telephone.type in self.telephoneTypes
101         set.add(self, telephone)
```

```
102
103 class TelephoneRecord:
104     """
105     Запись в телефонном справочнике
106     """
107     def __init__(self, telephone, collaborator, group):
108         self.telephone = telephone
109         self.collaborator = collaborator
110         self.group = group
111
112     def __hash__(self):
113         return hash((self.telephone, self.collaborator, self.group))
114
115     def __eq__(self, other):
116         return self.telephone == other.telephone and \
117             self.collaborator == other.collaborator and \
118             self.group == other.group
119
120 class TelephoneDir(set):
121     """
122     Телефонный справочник
123     """
124     def __init__(self, telephones, collaborators, groups):
125         set.__init__(self)
126         self.telephones = telephones
127         self.collaborators = collaborators
128         self.groups = groups
```

```
129
130 def add(self, telephoneRecord):
131     assert telephoneRecord.telephone in self.telephones
132     assert telephoneRecord.collaborator in self.collaborators
133     assert telephoneRecord.group in self.groups
134     assert telephoneRecord not in self
135     set.add(self, telephoneRecord)
136
137 if __name__ == '__main__':
138     import tdcsv
139
140     telephoneDir = tdcsv.load()
141
142     for r in telephoneDir:
143         if r.group.name == 'помощник проректора' and
r.collaborator.family.find('ск') >= 0:
144             print r.telephone.number, "%s %s. %s."% \
145                 (r.collaborator.family, r.collaborator.name[:2],
r.collaborator.patronym[:2])
146
147     for r in telephoneDir:
148         if r.group.name == 'зав. кафедрой' and r.collaborator.family.find('cc')
>= 0:
149             print r.telephone.number, "%s %s. %s."% \
150                 (r.collaborator.family, r.collaborator.name[:2],
r.collaborator.patronym[:2])
```

Заметно, что один и тот же сотрудник может быть членом различных групп. Поэтому работа запросов дала немного другой результат.

51-57-39 Виноградский С. Г.

51-18-84 Лосатинская А. С.

26-16-96 Коссович Л. Ю.

Сработал и второй запрос.

0.3 Набор заданий

0.3.1 Страховая компания

Описание предметной области

Вы работаете в страховой компании. Вашей задачей является отслеживание финансовой деятельности компании.

Компания имеет различные филиалы по всей стране. Каждый филиал характеризуется названием, адресом и телефоном. Деятельность компании организована следующим образом: к Вам обращаются различные лица с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков, договор заключается по определенному виду страхования (например, страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора Вы фиксируете дату заключения, страховую сумму, вид страхования, тарифную ставку и филиал, в котором заключался договор.

Классы объектов

Договоры (Номер договора, Дата заключения, Страховая сумма, Тарифная ставка, Филиал, Вид страхования).

Вид страхования (Вид страхования, Наименование).

Филиал (Филиал, Наименование филиала, Адрес, Телефон).

Развитие постановки задачи

Нужно учесть, что договоры заключают страховые агенты. Помимо информации об агентах (фамилия, имя, отчество, адрес, телефон), нужно еще хранить филиал, в котором работают агенты. Кроме того, исходя из базы данных, нужно иметь возможность рассчитывать заработную плату агентам. Заработная плата составляет некоторый процент от страхового платежа (страховой платеж это страховая сумма, умноженная на тарифную ставку). Процент зависит от вида страхования, по которому заключен договор.

0.3.2 Гостиница

Описание предметной области

Вы работаете в гостинице. Вашей задачей является отслеживание финансовой стороны работы гостиницы.

Ваша деятельность организована следующим образом: гостиница предоставляет номера клиентам на определенный срок. Каждый номер характеризуется вместимостью, комфортностью (люкс, полулюкс, обычный) и ценой. Вашими клиентами являются различные лица, о которых Вы собираете определенную информацию (фамилия, имя, отчество и некоторый комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При поселении фиксируется дата поселения. При выезде из гостиницы для каждого места запоминается дата освобождения.

Классы объектов

Клиенты (Клиент, Фамилия, Имя, Отчество, Паспортные данные, Комментарий).

Номера (Номер, Количество человек, Комфортность, Цена).

Поселение (Клиент, Номер, Дата поселения, Дата освобождения, Примечание).

Развитие постановки задачи

Необходимо хранить информацию не только по факту сдачи номера клиенту, но и осуществлять бронирование номеров. Кроме того, для постоянных клиентов, а также для определенных категорий клиентов, предусмотрена система скидок. Скидки могут суммироваться.

Внести в структуру сущностей изменения, учитывающие этот факт, и изменить существующие запросы. Добавить новые запросы.

0.3.3 Ломбард

Описание предметной области

Вы работаете в ломбарде. Вашей задачей является отслеживание финансовой стороны работы ломбарда.

Деятельность Вашей компании организована следующим образом: к Вам обращаются различные лица с целью получения денежных средств под залог определенных товаров. У каждого из приходящих к Вам клиентов Вы запрашиваете фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара Вы определяете сумму, которую готовы выдать на руки клиенту, а также свои комиссионные. Кроме того, определяете срок возврата денег. Если клиент согласен, то Ваши договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается у Вас. В случае если в указанный срок не происходит возврата денег, товар переходит в Вашу собственность.

Классы объектов

Клиенты (Клиент, Фамилия, Имя, Отчество, Номер паспорта, Серия паспорта, Дата выдачи паспорта).

Категории товаров (Категория товаров, Название, Примечание).

Сдача в ломбард (Категория товаров, Клиент, Описание товара, Дата сдачи, Дата возврата, Сумма, Комиссионные).

Развитие постановки задачи

После перехода прав собственности на товар, ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке. (Например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы). Помимо текущей цены, нужно хранить все возможные значения цены для данного товара.

0.3.4 Реализация готовой продукции

Описание предметной области

Вы работаете в компании, занимающейся оптово-розничной продажей различных товаров. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность Вашей компании организована следующим образом: Ваша компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В Вашу компанию обращаются покупатели. Для каждого из них Вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с покупателем количество купленного им товара и дату покупки.

Классы объектов

Товары (Наименование, Оптовая цена, Розничная цена, Описание).

Покупатели (Телефон, Контактное лицо, Адрес).

Сделки (Дата сделки, Товар, Количество, Покупатель, Признак оптовой продажи).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что обычно покупатели в рамках одной сделки покупают не один товар, а сразу несколько. Также компания решила предоставлять скидки в зависимости от количества закупленных товаров и их общей стоимости.

0.3.5 Ведение заказов

Описание предметной области

Вы работаете в компании, занимающейся оптовой продажей различных товаров. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность Вашей компании организована следующим образом: Ваша компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В Вашу компанию обращаются заказчики. Для каждого из них Вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с заказчиком количество купленного им товара и дату покупки.

Классы объектов

Заказчики (Наименование, Адрес, Телефон, Контактное лицо).

Товары (Цена, Доставка, Описание).

Заказы (Заказчик, Товар, Количество, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что доставка разных товаров может производиться разными способами, различными по цене и скорости. Нужно хранить информацию по тому, какими способами может осуществляться доставка каждого товара и информацию о том, какой вид доставки (а, соответственно, и какую стоимость доставки) выбрал клиент при заключении сделки.

0.3.6 Бюро по трудоустройству

Описание предметной области

Вы работаете в бюро по трудоустройству. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность Вашего бюро организована следующим образом: Ваше бюро готово искать работников для различных работодателей и вакансии для ищущих работу специалистов различного профиля. При обращении к Вам клиента-работодателя, его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. При обращении к Вам клиента-соискателя, его стандартные данные (фамилия, имя, отчество, квалификация, профессия, иные данные) также фиксируются в базе данных. По каждому факту удовлетворения интересов обеих сторон составляется документ. В документе указываются соискатель, работодатель, должность и комиссионные (доход бюро).

Классы объектов

Работодатели (Название, Вид деятельности, Адрес, Телефон). *Сделки* (Работодатель, Должность, Комиссионные). *Соискатели* (Фамилия, Имя, Отчество, Квалификация, Вид деятельности, Иные данные, Предполагаемый размер заработной платы).

Развитие постановки задачи

Оказалось, что база данных не совсем точно описывает работу бюро. В базе фиксируется только сделка, а информация по открытым вакансиям не храниться. Кроме того, для автоматического поиска вариантов, необходимо вести справочник «виды деятельности».

0.3.7 Нотариальная контора

Описание предметной области

Вы работаете в нотариальной конторе. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность Вашей нотариальной конторы организована следующим образом: Ваша фирма готова предоставить клиенту определенный комплекс услуг. Для наведения порядка Вы формализовали эти услуги, составив их список с описанием каждой услуги. При обращении к Вам клиента, его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ. В документе указываются услуга, сумма сделки, комиссионные (доход конторы), описание сделки.

Классы объектов

Клиенты (Название, Вид деятельности, Адрес, Телефон).

Сделки (Клиент, Услуга, Сумма, Комиссионные, Описание).

Услуги (Название, Описание).

Развитие постановки задачи

Теперь ситуация изменилась. В рамках одной сделки клиенту может быть оказано несколько услуг. Стоимость каждой услуги фиксирована. Кроме того, компания предоставляет в рамках одной сделки различные виды скидок. Скидки могут суммироваться.

Фирма по продаже запчастей

Описание предметной области

Вы работаете в фирме, занимающейся продажей запасных частей для автомобилей. Вашей задачей является отслеживание финансовой стороны работы компании.

Основная часть деятельности, находящейся в Вашем ведении, связана с работой с поставщиками. Фирма имеет определенный набор поставщиков, по каждому из которых известны название, адрес и телефон. У этих поставщиков Вы приобретаете детали. Каждая деталь наряду с названием характеризуется артикулом и ценой (считаем цену постоянной). Некоторые из поставщиков могут поставлять одинаковые детали (один и тот же артикул). Каждый факт покупки запчастей у поставщика фиксируется в базе данных, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей.

Классы объектов

Поставщики (Поставщик, Название, Адрес, Телефон).

Детали (Название, Артикул, Цена, Примечание).

Поставки (Поставщик, Деталь, Количество, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что цена детали может меняться от поставки к поставке. Поставщики заранее ставят Вас в известность о дате изменения цены и о его новом значении. Нужно хранить не только текущее значение цены, но и всю историю изменения цен.

0.3.8 Курсы по повышению квалификации

Описание предметной области

Вы работаете в учебном заведении и занимаетесь организацией курсов повышения квалификации.

В Вашем распоряжении имеются сведения о сформированных группах студентов. Группы формируются в зависимости от специальности и отделения. В каждой из них включено определенное количество студентов. Проведение занятий обеспечивает штат преподавателей. Для каждого из них у Вас в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки Вы получаете информацию о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Кроме того, хранятся также сведения о виде проводимых занятий (лекции, практика), предмете и оплате за 1 час.

Классы объектов

Группы (Специальность, Отделение, Количество студентов).

Преподаватели (Фамилия, Имя, Отчество, Телефон, Стаж).

Нагрузка (Преподаватель, Группа, Количество часов, Предмет, Тип занятия, Оплата).

Развитие постановки задачи

В результате работы с базой данных выяснилось, что размер почасовой оплаты зависит от предмета и типа занятия. Кроме того, каждый преподаватель может вести не все предметы, а только некоторые.

0.3.9 Определение факультативов для студентов

Описание предметной области

Вы работаете в высшем учебном заведении и занимаетесь организацией факультативов.

В Вашем распоряжении имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Преподаватели Вашей кафедры должны обеспечить проведение факультативных занятий по некоторым предметам. По каждому факультативу существует определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы). В результате работы со студентами у Вас появляется информация о том, кто из них записался на какие факультативы. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра Вы заносите информацию об оценках, полученных студентами на экзаменах.

Классы объектов

Студенты (Фамилия, Имя, Отчество, Адрес, Телефон).

Предметы (Название, Объем лекций, Объем практик, Объем лабораторных работ).

Учебный план (Студент, Предмет, Оценка).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что некоторые из факультативов могут длиться более одного семестра. В каждом семестре для предмета устанавливается объем лекций, практик и лабораторных работ в часах. В качестве итоговой оценки за предмет берется последняя оценка, полученная студентом.

0.3.10 Распределение учебной нагрузки

Описание предметной области

Вы работаете в высшем учебном заведении и занимаетесь распределением нагрузки между преподавателями кафедры.

В Вашем распоряжении имеются сведения о преподавателях кафедры, включающие наряду с анкетными данными сведения об их ученой степени, занимаемой административной должности и стаже работы. Преподаватели Вашей кафедры должны обеспечить проведение занятий по некоторым предметам. По каждому из них существует определенное количество часов. В результате распределения нагрузки у Вас должна получиться информация следующего рода: «Такой-то преподаватель проводит занятия по такому-то предмету с такой-то группой».

Классы объектов

Преподаватели (Фамилия, Имя, Отчество, Ученая степень, Должность, Стаж).

Предметы (Название, Количество часов).

Нагрузка (Преподаватель, Предмет, Номер группы).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что все проводимые занятия делятся на лекционные и практические. По каждому виду занятий устанавливается свое количество часов. Кроме того, данные по нагрузке нужно хранить несколько лет.

0.3.11 Распределение дополнительных обязанностей

Описание предметной области

Вы работаете в коммерческой компании и занимаетесь распределением дополнительных разовых работ. Вашей задачей является отслеживание хода выполнения дополнительных работ.

Компания имеет определенный штат сотрудников, каждый из которых получает определенный оклад. Время от времени, возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных должностных обязанностей сотрудников. Для наведения порядка в этой сфере деятельности Вы проклассифицировали все виды дополнительных работ, определившись с суммой оплаты по факту их выполнения. При возникновении дополнительной работы определенного вида Вы назначаете ответственного, фиксируя дату начала. По факту окончания Вы фиксируете дату и выплачиваете дополнительную сумму к зарплате с учетом Вашей классификации.

Классы объектов

Сотрудники (Фамилия, Имя, Отчество, Оклад).

Виды работ (Описание, Оплата за день).

Работы (Сотрудник, Вид работ, Дата начала, Дата окончания).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что некоторые из дополнительных работ являются достаточно трудоемкими и, в то же время, срочными, что требует привлечения к их выполнению нескольких сотрудников. Также оказалось, что длительность работ в каждом конкретном случае

составляет разную величину. Соответственно, нужно заранее планировать длительность работы и количество сотрудников, занятых для выполнения работы.

0.3.12 Техническое обслуживание станков

Описание предметной области

Ваше предприятие занимается ремонтом станков и другого промышленного оборудования. Вашей задачей является отслеживание финансовой стороны деятельности предприятия.

Клиентами Вашей компании являются промышленные предприятия, оснащенные различным сложным оборудованием. В случае поломок оборудования они обращаются к Вам.

Ремонтные работы в Вашей компании организованы следующим образом: все станки проклассифицированы по странам-производителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта Вы фиксируете вид станка и дату начала ремонта.

Классы объектов

Виды станков (Страна, Год выпуска, Марка).

Виды ремонта (Название, Продолжительность, Стоимость, Примечания).

Ремонт (Вид станка, Ремонт, Дата начала, Примечания).

Развитие постановки задачи

Теперь ситуация изменилась. Несложный анализ показал, что нужно не просто подразделять станки по типам, а иметь информацию о том, сколько раз ремонтировался тот или иной конкретный станок.

0.3.13 Туристическая фирма

Описание предметной области

Вы работаете в туристической компании. Ваша компания работает с клиентами, продавая им путевки. Вашей задачей является отслеживание финансовой стороны деятельности фирмы.

Работа с клиентами в Вашей компании организована следующим образом: у каждого клиента, пришедшего к Вам, собираются некоторые стандартные данные – фамилия, имя, отчество, адрес, телефон. После этого Ваши сотрудники выясняют у клиента, куда он хотел бы поехать отдыхать. При этом ему демонстрируются различные варианты, включающие страну проживания, особенности местного климата, имеющиеся отели разного класса. Наряду с этим, обсуждается возможная длительность пребывания и стоимость путевки. В случае если удалось договориться, и найти для клиента приемлемый вариант, Вы регистрируете факт продажи путевки (или путевок, если клиент покупает сразу несколько путевок), фиксируя дату отправления. Иногда Вы решаете предоставить клиенту некоторую скидку.

Классы объектов

Маршруты (Страна, Климат, Длительность, Отель, Стоимость).

Путевки (Маршрут, Клиент, Дата отправления, Количество, Скидка).

Клиенты (Фамилия, Имя, Отчество, Адрес, Телефон).

Развитие постановки задачи

Теперь ситуация изменилась. Фирма работает с несколькими отелями в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и

отеля. Скидки, которые предоставляет фирма, фиксированы. Например, при покупке более 1 путевки, предоставляется скидка 5

0.3.14 Грузовые перевозки

Описание предметной области

Вы работаете в компании, занимающейся перевозками грузов. Вашей задачей является отслеживание стоимости перевозок с учетом заработной платы водителей.

Ваша компания осуществляет перевозки по различным маршрутам. Для каждого маршрута Вы определили некоторое название, вычислили примерное расстояние и установили некоторую оплату для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов Вы храните полную информацию о перевозках (маршрут, водитель, даты отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия.

Классы объектов

Маршруты (Название, Дальность, Количество дней в пути, Оплата).

Водители (Фамилия, Имя, Отчество, Стаж).

Проделанная работа (Маршрут, Водитель, Дата отправки, Дата возвращения, Премия).

Развитие постановки задачи

Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему оплаты. Так, оплата водителям должна теперь зависеть не только от маршрута, но и от стажа водителя. Кроме того, нужно учесть, что перевозки могут осуществлять два водителя.

0.3.15 Учет телефонных переговоров

Описание предметной области

Вы работаете в коммерческой службе телефонной компании. Компания предоставляет абонентам телефонные линии для междугородних переговоров. Вашей задачей является отслеживание стоимости междугородних телефонных переговоров.

Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток.

Классы объектов

Абоненты (Номер телефона, ИНН, Адрес).

Города (Название, Тариф дневной, Тариф ночной).

Переговоры (Абонент, Город, Дата, Количество минут, Время суток).

Развитие постановки задачи

Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему скидок. Так, стоимость минуты теперь уменьшается в зависимости от длительности разговора. Размер скидки для каждого города разный.

0.3.16 Учет внутриофисных расходов

Описание предметной области

Вы работаете в бухгалтерии частной фирмы. Сотрудники фирмы имеют возможность осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Вашей задачей является отслеживание внутриофисных расходов.

Ваша фирма состоит из отделов. Каждый отдел имеет название. В каждом отделе работает определенное количество сотрудников. Сотрудники могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены по данному виду расходов в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел.

Классы объектов

Отделы (Название, Количество сотрудников).

Виды расходов (Название, Описание, Предельная норма).

Расходы (Вид расходов, Отдел, Сумма, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Оказалось, что нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Неиспользованные в текущем месяце деньги могут быть использованы позже.

0.3.17 Библиотека

Описание предметной области

Вы являетесь руководителем библиотеки. Ваша библиотека решила зарабатывать деньги, выдавая напрокат некоторые книги, имеющиеся в небольшом количестве экземпляров. Вашей задачей является отслеживание финансовых показателей работы библиотеки.

У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги Вы определили для каждой из них залоговую стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). В библиотеку обращаются читатели. Все читатели регистрируются в картотеке, которая содержит стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Каждый читатель может обращаться в библиотеку несколько раз. Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата.

Классы объектов

Книги (Название, Автор, Залоговая стоимость, Стоимость проката, Жанр).

Читатели (Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные книги (Книга, Читатель, Дата выдачи, Дата возврата).

Развитие постановки задачи

Теперь ситуация изменилась. Несложный анализ показал, что стоимость проката книги должна зависеть не только от самой книги, но и от срока ее проката. Кроме того, необходимо добавить систему штрафов за вред, нанесенный книге и систему скидок для некоторых категорий читателей.

0.3.18 Прокат автомобилей

Описание предметной области

Вы являетесь руководителем коммерческой службы в фирме, занимающейся прокатом автомобилей. Вашей задачей является отслеживание финансовых показателей работы пункта проката.

В Ваш автопарк входит некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката. В пункт проката обращаются клиенты. Все клиенты проходят обязательную регистрацию, при которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Каждый клиент может обращаться в пункт проката несколько раз. Все обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата.

Классы объектов

Автомобили (Марка, Стоимость, Стоимость проката, Тип).

Клиенты (Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные автомобили (Автомобиль, Клиент, Дата выдачи, Дата возврата).

Развитие постановки задачи

Теперь ситуация изменилась. Несложный анализ показал, что стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от срока его проката, а также от года выпуска. Также нужно ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов.

0.3.19 Выдача банком кредитов

Описание предметной области

Вы являетесь руководителем информационно-аналитического центра коммерческого банка. Одним из существенных видов деятельности Вашего банка является выдача кредитов юридическим лицам. Вашей задачей является отслеживание динамики работы кредитного отдела.

В зависимости от условий получения кредита, процентной ставки и срока возврата все кредитные операции делятся на несколько основных видов. Каждый из этих видов имеет свое название. Кредит может получить юридическое лицо (клиент), при регистрации предоставивший следующие сведения: название, вид собственности, адрес, телефон, контактное лицо. Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, клиент и дата выдачи.

Классы объектов

Виды кредитов (Название, Условия получения, Ставка, Срок).

Клиенты (Название, Вид собственности, Адрес, Телефон, Контактное лицо).

Кредиты (Вид кредитов, Клиент, Сумма, Дата выдачи).

Развитие постановки задачи

Теперь ситуация изменилась. После проведения различных исследований выяснилось, что используемая система не позволяет отслеживать динамику возврата кредитов. Для устранения этого недостатка Вы приняли решение учитывать в системе еще и дату фактического возврата денег. Нужно еще учесть, что кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы.

0.3.20 Инвестирование свободных средств

Описание предметной области

Вы являетесь руководителем аналитического центра инвестиционной компании. Ваша компания занимается вложением денежных средств в ценные бумаги.

Ваши клиенты – предприятия, которые доверяют Вам управлять их свободными денежными средствами на определенный период. Вам необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и Вам и Вашему клиенту. При работе с клиентом для Вас весьма существенной является информация о предприятии – название, вид собственности, адрес и телефон.

Классы объектов

Ценные бумаги (Код ценной бумаги, Минимальная сумма сделки, Рейтинг, Доходность за прошлый год, Дополнительная информация).

Инвестиции (Ценная бумага, Клиент, Котировка, Дата покупки, Дата продажи).

Клиенты (Клиент, Название, Вид собственности, Адрес, Телефон).

Развитие постановки задачи

При эксплуатации базы данных стало понятно, что необходимо хранить историю котировок каждой ценной бумаги. Кроме того, помимо вложений в ценные бумаги, существует возможность вкладывать деньги в банковские депозиты.

0.3.21 Занятость актеров театра

Описание предметной области

Вы являетесь коммерческим директором театра, и в Ваши обязанности входит вся организационно-финансовая работа, связанная с привлечением актеров и заключением контрактов.

Вы поставили дело следующим образом: каждый год театр осуществляет постановку различных спектаклей. Каждый спектакль имеет определенный бюджет. Для участия в конкретных постановках в определенных ролях Вы привлекаете актеров. С каждым из актеров Вы заключаете персональный контракт на определенную сумму. Каждый из актеров имеет некоторый стаж работы, некоторые из них удостоены различных наград и званий.

Классы объектов

Актеры (Фамилия, Имя, Отчество, Звание, Стаж).

Спектакли (Название, Год постановки, Бюджет).

Занятость актеров в спектакле (Актер, Спектакль, Роль, Стоимость годового контракта).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что в рамках одного спектакля на одну и ту же роль привлекается несколько актеров. Контракт определяет базовую зарплату актера, а по итогам реально отыгранных спектаклей актеру назначается премия. Кроме того, в базе данных нужно хранить информацию за несколько лет.

0.3.22 Платная поликлиника

Описание предметной области

Вы являетесь руководителем службы планирования платной поликлиники. Вашей задачей является отслеживание финансовых показателей работы поликлиники.

В поликлинике работают врачи различных специальностей, имеющие разную квалификацию. Каждый день в поликлинику обращаются больные. Все больные проходят обязательную регистрацию, при которой в базу данных заносятся стандартные анкетные данные (фамилия, имя, отчество, год рождения). Каждый больной может обращаться в поликлинику несколько раз, нуждаясь в различной медицинской помощи. Все обращения больных фиксируются, при этом устанавливается диагноз, определяется стоимость лечения, запоминается дата обращения.

Классы объектов

Врачи (Фамилия, Имя, Отчество, Специальность, Категория).

Пациенты (Фамилия, Имя, Отчество, Год рождения).

Обращения (Врач, Пациент, Дата обращения, Диагноз, Стоимость лечения).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что при обращении в поликлинику пациент обследуется и проходит лечение у разных специалистов. Общая стоимость лечения зависит от стоимости тех консультаций и процедур, которые назначены пациенту. Кроме того, для определенных категорий граждан предусмотрены скидки.

0.3.23 Анализ динамики показателей финансовой отчетности различных предприятий

Описание предметной области

Вы являетесь руководителем информационно-аналитического центра крупного холдинга. Вашей задачей является отслеживание динамики показателей для предприятий Вашего холдинга.

В структуру холдинга входят несколько предприятий. Каждое предприятие имеет стандартные характеристики (название, реквизиты, телефон, контактное лицо). Работа предприятия может быть оценена следующим образом: в начале каждого отчетного периода на основе финансовой отчетности вычисляется по неким формулам определенный набор показателей. Принять, что важность показателей характеризуется некоторыми числовыми константами. Значение каждого показателя измеряется в некоторой системе единиц.

Классы объектов

Показатели (Название, Важность, Единица измерения).

Предприятия (Название, Банковские реквизиты, Телефон, Контактное лицо).

Динамика показателей (Показатель, Предприятие, Дата, Значение).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что некоторые показатели считаются в рублях, некоторые в долларах, некоторые в евро. Для удобства работы с показателями нужно хранить изменения курсов валют относительно друг друга.

0.3.24 Учет телекомпанией стоимости прошедшей в эфире рекламы

Описание предметной области

Вы являетесь руководителем коммерческой службы телевизионной компании. Вашей задачей является отслеживание расчетов, связанных с прохождением рекламы в телеэфире.

Работа построена следующим образом: заказчики просят поместить свою рекламу в определенной передаче в определенный день. Каждый рекламный ролик имеет определенную продолжительность. Для каждой организации-заказчика известны банковские реквизиты, телефон и контактное лицо для проведения переговоров. Передачи имеют определенный рейтинг. Стоимость минуты рекламы в каждой конкретной передаче известна (определяется коммерческой службой, исходя из рейтинга передачи и прочих соображений).

Классы объектов

Передачи (Название, Рейтинг, Стоимость минуты).

Реклама (Передача, Заказчик, Дата, Длительность в минутах).

Заказчики (Название, Банковские реквизиты, Телефон, Контактное лицо).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что необходимо также хранить информацию об агентах, заключивших договоры на рекламу. Зарплата рекламных агентов составляет некоторый процент от общей стоимости рекламы, прошедшей в эфире.

0.3.25 Интернет-магазин

Описание предметной области

Вы являетесь сотрудником коммерческого отдела компании, продающей различные товары через Интернет. Вашей задачей является отслеживание финансовой составляющей работы компании.

Работа Вашей компании организована следующим образом: на Интернет-сайте компании представлены (выставлены на продажу) некоторые товары. Каждый из них имеет некоторое название, цену и единицу измерения (штуки, килограммы, литры). Для проведения исследований и оптимизации работы магазина Вы пытаетесь собирать данные с Ваших клиентов. При этом для Вас определяющее значение имеют стандартные анкетные данные, а также телефон и адрес электронной почты для связи. В случае приобретения товаров на сумму свыше 5000р. клиент переходит в категорию «постоянных клиентов» и получает скидку на каждую покупку в размере 2%. По каждому факту продажи Вы автоматически фиксируете клиента, товары, количество, дату продажи, дату доставки.

Классы объектов

Товары (Название, Цена, Единица измерения).

Клиенты (Фамилия, Имя, Отчество, Адрес, Телефон, email, Признак постоянного клиента).

Продажи (Товар, Клиент, Дата продажи, Дата доставки, Количество).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что иногда возникают проблемы, связанные с нехваткой информации о наличии нужных товаров на складе в нужном количестве. Кроме того,

обычно клиенты в рамках одного заказа покупают не один вид товара, а несколько видов. Исходя из суммарной стоимости заказа, компания предоставляет дополнительные скидки.

0.3.26 Ювелирная мастерская

Описание предметной области

Вы работаете в ювелирной мастерской. Ваша мастерская осуществляет изготовление ювелирных изделий для частных лиц на заказ. Вы работаете с определенными материалами (платина, золото, серебро, различные драгоценные камни и т.д.). При обращении к Вам потенциального клиента Вы определяетесь с тем, какое именно изделие ему необходимо. Все изготавливаемые Вами изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), бывают выполнены из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы).

Классы объектов

Изделия (Название, Тип, Материал, Вес, Цена).

Материалы (Название, Цена за грамм).

Продажи (Изделие, Дата продажи, Фамилия покупателя, Имя покупателя, Отчество покупателя).

Развитие постановки задачи

В процессе опытной эксплуатации базы данных выяснилось, что ювелирное изделие может состоять из нескольких материалов. Кроме того, постоянным клиентам мастерская предоставляет скидки.

0.3.27 Парикмахерская

Описание предметной области

Вы работаете в парикмахерской.

Ваша парикмахерская стрижет клиентов в соответствии с их пожеланиями и некоторым каталогом различных видов стрижки. Так, для каждой стрижки определены название, принадлежность полу (мужская, женская), стоимость работы. Для наведения порядка Вы, по мере возможности, составляете базу данных клиентов, запоминая их анкетные данные (фамилия, имя, отчество). Начиная с 5-ой стрижки, клиент переходит в категорию постоянных и получает скидку в 3% при каждой последующей стрижке. После того, как закончена очередная работа, в кассе фиксируются стрижка, клиент и дата производства работ.

Классы объектов

Стрижки (Название, Пол, Стоимость).

Клиенты (Фамилия, Имя, Отчество, Пол, Признак постоянного клиента).

Работа (Стрижка, Клиент, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. У Вашей парикмахерской появился филиал, и Вы хотели бы видеть, в том числе, и отдельную статистику по филиалам. Кроме того, стоимость стрижки может меняться с течением времени. Нужно хранить не только последнюю цену, но и все данные по изменению цены стрижки.

0.3.28 Химчистка

Описание предметной области

Вы работаете в химчистке.

Ваша химчистка осуществляет прием у населения вещей для выведения пятен. Для наведения порядка Вы, по мере возможности, составляете базу данных клиентов, запоминая их анкетные данные (фамилия, имя, отчество). Начиная с 3-го обращения, клиент переходит в категорию постоянных клиентов и получает скидку в 3% при чистке каждой последующей вещи. Все оказываемые Вами услуги подразделяются на виды, имеющие название, тип и стоимость, зависящую от сложности работ. Работа с клиентом первоначально состоит в определении объема работ, вида услуги и, соответственно, ее стоимости. Если клиент согласен, он оставляет вещь (при этом фиксируется услуга, клиент и дата приема) и забирает ее после обработки (при этом фиксируется дата возврата).

Классы объектов

Виды услуг (Название, Тип, Стоимость).

Клиенты (Фамилия, Имя, Отчество, Признак постоянного клиента).

Услуги (Вид услуги, Клиент, Дата приема, Дата возврата).

Развитие постановки задачи

Теперь ситуация изменилась. У Вашей химчистки появился филиал, и Вы хотели бы видеть, в том числе, и отдельную статистику по филиалам. Кроме того, вы решили делать надбавки за срочность и сложность работ.

0.3.29 Сдача в аренду торговых площадей

Описание предметной области

Вы работаете в крупном торговом центре, сдающим в аренду коммерсантам свои торговые площади.

Вашей задачей является наведение порядка в финансовой стороне работы торгового центра.

Работы Вашего торгового центра построена следующим образом: в результате планирования Вы определили некоторое количество торговых точек в пределах Вашего здания, которые могут сдаваться в аренду. Для каждой из торговых точек важными данными являются этаж, площадь, наличие кондиционера и стоимость аренды в день. Со всех потенциальных клиентов Вы собираете стандартные данные (название, адрес, телефон, реквизиты, контактное лицо). При появлении потенциального клиента Вы показываете ему имеющиеся свободные площади. При достижении соглашения Вы оформляете договор, фиксируя в базе данных торговую точку, клиента, период (срок) аренды.

Классы объектов

Торговые точки (Этаж, Площадь, Наличие кондиционера, Стоимость аренды в день).

Клиенты (Название, Реквизиты, Адрес, Телефон, Контактное лицо).

Аренда (Торговая точка, Клиент, Дата начала, Дата окончания).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что некоторые клиенты арендуют сразу несколько торговых точек. Помимо этого, Вам необходимо собирать информацию об ежемесячных платежах, поступающих Вам от арендаторов.

1 Основные понятия технологии проектирования информационных систем

Информация в современном мире превратилась в один из наиболее важных ресурсов, а информационные системы (ИС) стали необходимым инструментом практически во всех сферах деятельности.

Разнообразие задач, решаемых с помощью ИС, привело к появлению множества разнотипных систем, отличающихся принципами построения и заложенными в них правилами обработки информации.

Информационные системы можно классифицировать по целому ряду различных признаков. В основу рассматриваемой классификации положены наиболее существенные признаки, определяющие функциональные возможности и особенности построения современных систем. В зависимости от объема решаемых задач, используемых технических средств, организации функционирования, информационные системы делятся на ряд групп (классов).

По типу данных: фактографические и документальные.

По степени автоматизации: ручные, автоматизированные и автоматические.

По сфере применения интегрированные, организационного управления, управления технологическим процессом и автоматизированного проектирования.

По характеру обработки информационно-поисковые и информационно-решающие (управляющие, советующие).

По уровню управления стратегические, функциональные и операционные.

По типу хранимых данных ИС делятся на фактографические и документальные.

Фактографические системы предназначены для хранения и обработки структурированных данных в виде чисел и текстов. Над такими данными можно выполнять различные операции.

В *документальных системах* информация представлена в виде документов, состоящих из наименований, описаний, рефератов и текстов. Поиск по неструктурированным данным осуществляется с использованием семантических признаков. Отобранные документы предоставляются пользователю, а обработка данных в таких системах практически не производится.

Основываясь на степени автоматизации информационных процессов в системе управления фирмой, информационные системы делятся на ручные, автоматические и автоматизированные.

Ручные ИС характеризуются отсутствием современных технических средств переработки информации и выполнением всех операций человеком.

Автоматизированные ИС предполагают участие в процессе обработки информации и человека, и технических средств, причем главная роль в выполнении рутинных операций обработки данных отводится компьютеру. Именно этот класс систем соответствует современному представлению понятия “информационная система”.

В *автоматических ИС* все операции по переработке информации выполняются без участия человека.

В зависимости от сферы применения различают следующие классы ИС.

ИС организационного управления – предназначены для автоматизации функций управленческого персонала как промышленных предприятий, так и непромышленных объектов (гостиниц, банков,

магазинов и пр.). Основными функциями подобных систем являются: оперативный контроль и регулирование, оперативный учет и анализ, перспективное и оперативное планирование, бухгалтерский учет, управление сбытом, снабжением и другие экономические и организационные задачи.

ИС управления технологическими процессами – служат для автоматизации функций производственного персонала по контролю и управлению производственными операциями. В таких системах обычно предусматривается наличие развитых средств измерения параметров технологических процессов (температуры, давления, химического состава и т.п.), процедур контроля допустимости значений параметров и регулирования технологических процессов.

ИС автоматизированного проектирования (САПР) – предназначены для автоматизации функций инженеров-проектировщиков, конструкторов, архитекторов, дизайнеров при создании новой техники или технологии. Основными функциями подобных систем являются: инженерные расчеты, создание графической документации (чертежей, схем, планов), создание проектной документации, моделирование проектируемых объектов.

В зависимости от характера обработки данных ИС делятся на информационно-поисковые и информационно-решающие.

Информационно-поисковые системы производят ввод, систематизацию, хранение, выдачу информации по запросу пользователя без сложных преобразований данных. (Например, ИС библиотечного обслуживания, резервирования и продажи билетов на транспорте, бронирования мест в гостиницах и пр.)

Информационно-решающие системы осуществляют, кроме того, операции переработки информации по определенному алгоритму. По характеру использования выходной информации такие системы принято делить на управляющие и советующие.

Результирующая информация *управляющих ИС* непосредственно трансформируется в принимаемые человеком решения. Для этих систем характерны задачи расчетного характера и обработка больших объемов данных. (Например, ИС планирования производства или заказов, бухгалтерского

учета.)

Советующие ИС вырабатывают информацию, которая принимается человеком к сведению и учитывается при формировании управленческих решений, а не инициирует конкретные действия. Эти системы имитируют интеллектуальные процессы обработки знаний, а не данных. (Например, экспертные системы.)

Существует классификация ИС в зависимости от уровня управления, на котором система используется.

Стратегическая ИС – компьютерная информационная система, обеспечивающая поддержку принятия решений по реализации стратегических перспективных целей развития организации.

Информационные системы стратегического уровня помогают высшему звену управленцев решать неструктурированные задачи, осуществлять долгосрочное планирование. Основная задача – сравнение происходящих во внешнем окружении изменений с существующим потенциалом фирмы. Они призваны создать общую среду компьютерной телекоммуникационной поддержки решений в неожиданно возникающих ситуациях.

ИС оперативного уровня – поддерживает исполнителей, обрабатывая данные о сделках и событиях (счета, накладные, зарплата, кредиты, поток сырья и материалов). Информационная система оперативного уровня является связующим звеном между фирмой и внешней средой.

Функциональные ИС – используются работниками среднего управленческого звена для мониторинга, контроля, принятия решений и администрирования. Основные функции этих информационных систем:

- сравнение текущих показателей с прошлыми;
- составление периодических отчетов за определенное время, а не выдача отчетов по текущим событиям, как на оперативном уровне;
- обеспечение доступа к архивной информации и т.д.

С точки зрения программно-аппаратной реализации можно выделить ряд типовых архитектур ИС.

Традиционные архитектурные решения основаны на использовании выделенных файл-серверов или серверов баз данных. Существуют также варианты архитектур корпоративных информационных систем, базирующихся на технологии Internet (Intranet-приложения). Следующая разновидность архитектуры информационной системы основывается на концепции “хранилища данных” (DataWarehouse) - интегрированной информационной среды, включающей разнородные информационные ресурсы. И, наконец, для построения глобальных распределенных информационных приложений используется архитектура интеграции информационно-вычислительных компонентов на основе объектно-ориентированного подхода.

Индустрия разработки автоматизированных информационных систем управления зародилась в 1950-х – 1960-х годах и к концу века приобрела вполне законченные формы.

На первом этапе основным подходом в проектировании ИС был метод “снизу-вверх”, когда система создавалась как набор приложений, наиболее важных в данный момент для поддержки деятельности предприятия. Основной целью этих проектов было не создание тиражируемых продуктов, а обслуживание текущих потребностей конкретного учреждения. Такой подход отчасти сохраняется и сегодня. В рамках “лоскутной автоматизации” достаточно хорошо обеспечивается поддержка отдельных функций, но практически полностью отсутствует стратегия развития комплексной системы автоматизации, а объединение функциональных подсистем превращается в самостоятельную и достаточно сложную проблему.

Создавая свои отделы и управления автоматизации, предприятия пытались “обустроиться” своими силами. Однако периодические изменения технологий работы и должностных инструкций, сложности, связанные с разными представлениями пользователей об одних и тех же данных, приводили к непрерывным доработкам программных продуктов для удовлетворения все новых и новых пожеланий отдельных работников. Как следствие – и работа программистов, и создаваемые ИС вызвали недовольство руководителей и пользователей системы.

Следующий этап связан с осознанием того факта, что существует потребность в достаточно стандартных программных средствах автоматизации деятельности различных учреждений и предприятий. Из всего спектра проблем разработчики выделили наиболее заметные: автоматизацию ведения бухгалтерского аналитического учета и технологических процессов. Системы начали проектироваться “сверху-вниз”, т.е. в предположении, что одна программа должна удовлетворять потребности многих пользователей.

Сама идея использования универсальной программы накладывает существенные ограничения на возможности разработчиков по формированию структуры базы данных, экранных форм, по выбору алгоритмов расчета. Заложенные “сверху” жесткие рамки не дают возможности гибко адаптировать систему к специфике деятельности конкретного предприятия: учесть необходимую глубину аналитического и производственно-технологического учета, включить необходимые процедуры обработки данных, обеспечить интерфейс каждого рабочего места с учетом функций и технологии работы конкретного пользователя. Решение этих задач требует серьезных доработок системы. Таким образом, материальные и временные затраты на внедрение системы и ее доводку под требования заказчика обычно значительно превышают запланированные показатели.

Согласно статистическим данным, собранным Standish Group (США), из 8380 проектов, обследованных в США в 1994 году, неудачными оказались более 30% проектов, общая стоимость которых превышала 80 миллиардов долларов. При этом оказались выполненными в срок лишь 16% от общего числа проектов, а перерасход средств составил 189% от запланированного бюджета.

В то же время, заказчики ИС стали выдвигать все больше требований, направленных на обеспечение возможности комплексного использования корпоративных данных в управлении и планировании своей деятельности.

Интегрированные (корпоративные) ИС – используются для автоматизации всех функций фирмы и охватывают весь цикл работ от планирования деятельности до сбыта продукции. Они включают в себя ряд модулей (подсистем), работающих в едином информационном пространстве и выполняющих

функции поддержки со- ответствующих направлений деятельности.

Анализ современного состояния рынка ИС показывает устойчивую тенденцию роста спроса на информационные системы организационного управления. Причем спрос продолжает расти именно на интегрированные системы управления. Автоматизация отдельной функции, например, бухгалтерского учета или сбыта готовой продукции, считается уже пройденным этапом для многих предприятий.

Таким образом, возникла насущная необходимость формирования новой методологии построения информационных систем.

Цель такой методологии заключается в регламентации процесса проектирования ИС и обеспечении управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой ИС, так и к характеристикам процесса разработки. Основными задачами, решению которых должна способствовать методология проектирования корпоративных ИС, являются следующие:

- обеспечивать создание корпоративных ИС, отвечающих целям и задачам организации, а также предъявляемым требованиям по автоматизации деловых процессов заказчика;
- гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта;
- поддерживать удобную дисциплину сопровождения, модификации и наращивания системы;
- обеспечивать преемственность разработки, т.е. использование в разрабатываемой ИС существующей информационной инфраструктуры организации (задела в области информационных технологий).

Внедрение методологии должно приводить к снижению сложности процесса создания ИС за счет полного и точного описания этого процесса, а также применения современных методов и технологий создания ИС на всем жизненном цикле ИС – от замысла до реализации.

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.

Проектирование информационных систем всегда начинается с определения цели проекта. В общем виде цель проекта можно определить как решение ряда взаимосвязанных задач, включающих в себя обеспечение на момент запуска системы и в течение всего времени ее эксплуатации:

- требуемой функциональности системы и уровня ее адаптивности к изменяющимся условиям функционирования;
- требуемой пропускной способности системы;
- требуемого времени реакции системы на запрос;
- безотказной работы системы;
- необходимого уровня безопасности;
- простоты эксплуатации и поддержки системы.

Согласно современной методологии, процесс создания ИС представляет собой процесс построения и последовательного преобразования ряда согласованных моделей на всех этапах жизненного

цикла (ЖЦ) ИС. На каждом этапе ЖЦ создаются специфичные для него модели – организации, требований к ИС, проекта ИС, требований к приложениям и т.д. Модели формируются рабочими группами команды проекта, сохраняются и накапливаются в репозитории проекта. Создание моделей, их контроль, преобразование и предоставление в коллективное пользование осуществляется с использованием специальных программных инструментов – CASE-средств.

Процесс создания ИС делится на ряд этапов (стадий), ограниченных некоторыми временными рамками и заканчивающихся выпуском конкретного продукта (моделей, программных продуктов, документации и пр.).

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. Чтобы специфицировать процесс создания ИС, отвечающей потребностям организации, нужно выяснить и четко сформулировать, в чем заключаются эти потребности. Для этого необходимо определить требования заказчиков к ИС и отобразить их на языке моделей в требования к разработке проекта ИС так, чтобы обеспечить соответствие целям и задачам организации.

Задача формирования требований к ИС является одной из наиболее ответственных, трудно формализуемых и наиболее дорогих и тяжелых для исправления в случае ошибки. Современные инструментальные средства и программные продукты позволяют достаточно быстро создавать ИС по готовым требованиям. Но зачастую эти системы не удовлетворяют заказчиков, требуют многочисленных доработок, что приводит к резкому удорожанию фактической стоимости ИС. Основной причиной такого положения является неправильное, неточное или неполное определение требований к ИС на этапе анализа.

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). В

неоднородной ИС могут работать несколько компьютеров на разных аппаратных платформах и под управлением различных операционных систем. Кроме выбора платформы, на этапе проектирования определяются следующие характеристики архитектуры:

- будет ли это архитектура “файл-сервер” или “клиент-сервер”;
- будет ли это 3-уровневая архитектура со следующими слоями: сервер, ПО промежуточного слоя (сервер приложений), клиентское ПО;
- будет ли база данных централизованной или распределенной. Если база данных будет распределенной, то какие механизмы поддержки согласованности и актуальности данных будут использоваться;
- будет ли база данных однородной, то есть, будут ли все серверы баз данных продуктами одного и того же производителя (например, все серверы только Oracle или все серверы только DB2 UDB). Если база данных не будет однородной, то какое ПО будет использовано для обмена данными между СУБД разных производителей (уже существующее или разработанное специально как часть проекта);.
- будут ли для достижения должной производительности использоваться параллельные серверы баз данных (например, Oracle Parallel Server, DB2 UDB и т.п.).

Этап проектирования завершается разработкой технического проекта ИС.

На этапе реализации осуществляется создание программного обеспечения системы, установка технических средств, разработка эксплуатационной документации.

Этап тестирования обычно оказывается распределенным во времени.

Контрольные вопросы

1. Какой тип данных обрабатывается в фактографических информационных системах?

Структурированные данные в виде текстов и чисел

Графические изображения

Документы, состоящие из наименований, описаний, рефератов и текстов

2. Для какого типа информационных систем характерны процедуры поиска данных без организации их сложной обработки?

Для информационно-поисковых систем

Для информационных систем управления технологическими процессами

Для информационно-решающих систем

3. Какие функции реализуются в информационных системах организационного управления?

Измерение параметров технологических процессов

Контроль и управление производственными операциями

Инженерные расчеты

Оперативный учет

Перспективное и оперативное планирование

4. Какие из перечисленных функций реализуются в подсистеме маркетинга корпоративной ИС?

Анализ и установление цены

Финансовый анализ и прогнозирование
Анализ и планирование подготовки кадров
Анализ работы оборудования
Управление продажами

5. Какие из перечисленных функций реализуются в производственных подсистемах корпоративной ИС?

Планирование объемов работ и разработка календарных планов
Анализ и планирование подготовки кадров
Анализ работы оборудования
Управление продажами
Управление портфелем заказов

6. Какие из перечисленных функций реализуются в финансовых подсистемах корпоративной ИС?

Управление портфелем заказов
Управление запасами
Бухгалтерский учет и расчет зарплаты
Контроль бюджета
Управление продажами

7. Сформулируйте цель методологии проектирования ИС

Регламентация процесса проектирования ИС и обеспечение управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой ИС, так и к характеристикам процесса разработки

Автоматизация ведения бухгалтерского аналитического учета и технологических процессов
Формирование требований, направленных на обеспечение возможности комплексного использования корпоративных данных в управлении и планировании деятельности предприятия

8. Решению каких задач способствует внедрение методологии проектирования ИС?

Гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта

Обеспечить удобную дисциплину сопровождения, модификации и наращивания системы

Обеспечить нисходящее проектирование ИС (проектирование «сверху-вниз», в предположении, что одна программа должна удовлетворять потребности многих пользователей)

9. Укажите составляющие этапа проектирования ИС

Спецификация требований к приложениям

Инсталляция базы данных

Проектирование объектов данных

Выбор архитектуры ИС

Разработка программного кода приложений

Набрано баллов

2 Жизненный цикл программного обеспечения ИС

Жизненный цикл ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования.

В настоящее время известны и используются следующие модели жизненного цикла:

- *Каскадная модель* (рис. 2.1) предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.
- *Поэтапная модель* с промежуточным контролем (рис. 2.2). Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.
- *Спиральная модель* (рис. 2.3). На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Особое внимание уделяется начальным этапам разработки - анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования).



Рис. 2.1: Каскадная модель ЖЦ ИС

На практике наибольшее распространение получили две основные модели жизненного цикла: каскадная модель и спиральная модель.

Можно выделить следующие положительные стороны применения каскадного подхода:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении относительно простых или хорошо известных разработчикам ИС, когда в самом начале разработки можно достаточно точно и полно сформулировать все требования к системе. Основным недостатком этого подхода является то, что



Рис. 2.2: Поэтапная модель с промежуточным контролем

реальный процесс создания системы никогда полностью не укладывается в такую жесткую схему, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ИС оказывается соответствующим поэтапной модели с промежуточным контролем.

Спиральная модель ЖЦ была предложена для преодоления перечисленных проблем. На этапах анализа и проектирования реализуемость технических решений и степень удовлетворения потребностей заказчика проверяется путем создания прототипов. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы. Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который удовлетворяет действительным требованиям заказчика и доводится до реализации.

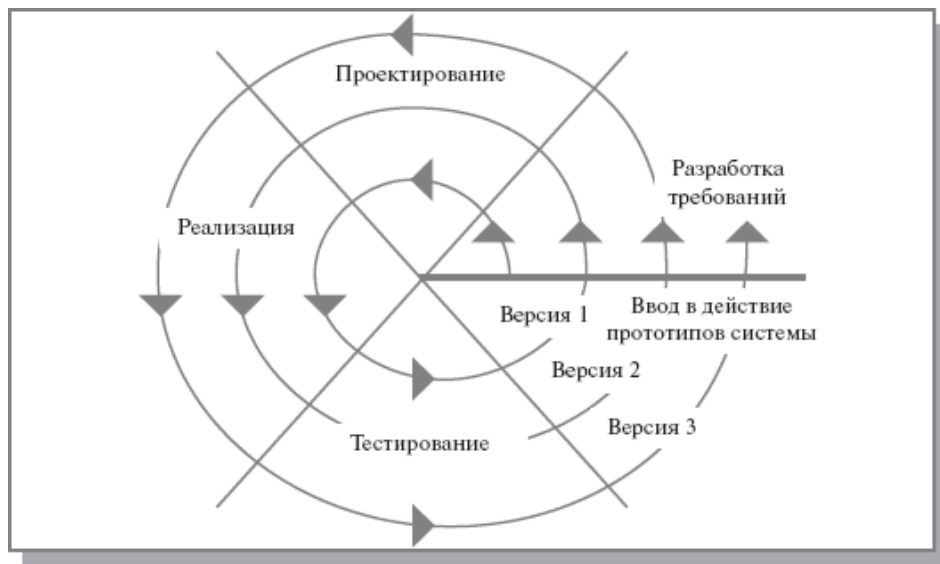


Рис. 2.3: Спиральная модель ЖЦ ИС

Итеративная разработка отражает объективно существующий спиральный цикл создания сложных систем. Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем и решить главную задачу - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения вводятся временные ограничения на каждый из этапов жизненного цикла, и переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. Планирование производится на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Существует целый ряд стандартов, регламентирующих ЖЦ ПО, а в некоторых случаях и процессы разработки. Среди наиболее известных стандартов можно выделить следующие:

- ГОСТ 34.601-90 - распространяется на автоматизированные системы и устанавливает стадии и этапы их создания. Кроме того, в стандарте содержится описание содержания работ на каждом этапе. Стадии и этапы работы, закрепленные в стандарте, в большей степени соответствуют каскадной модели жизненного цикла.
- ISO/IEC 12207:1995 - стандарт на процессы и организацию жизненного цикла. Распространяется на все виды заказного ПО. Стандарт не содержит описания фаз, стадий и этапов.
- Custom Development Method (методика Oracle) по разработке прикладных информационных систем - технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle. Применяется CDM для классической модели ЖЦ (предусмотрены все работы/задачи и этапы), а также для технологий “быстрой разработки” (Fast Track) или “облегченного подхода”, рекомендуемых в случае малых проектов.

- Rational Unified Process (RUP) предлагает итеративную модель разработки, включающую четыре фазы: начало, исследование, построение и внедрение. Каждая фаза может быть разбита на этапы (итерации), в результате которых выпускается версия для внутреннего или внешнего использования. Прохождение через четыре основные фазы называется циклом разработки, каждый цикл завершается генерацией версии системы. Если после этого работа над проектом не прекращается, то полученный продукт продолжает развиваться и снова минует те же фазы. Суть работы в рамках RUP - это создание и сопровождение моделей на базе UML.
- Microsoft Solution Framework (MSF) сходна с RUP, так же включает четыре фазы: анализ, проектирование, разработка, стабилизация, является итерационной, предполагает использование объектно-ориентированного моделирования. MSF в сравнении с RUP в большей степени ориентирована на разработку бизнес-приложений.
- Extreme Programming (XP). Экстремальное программирование (самая новая среди рассматриваемых методологий) сформировалось в 1996 году. В основе методологии командная работа, эффективная коммуникация между заказчиком и исполнителем в течение всего проекта по разработке ИС, а разработка ведется с использованием последовательно дорабатываемых прототипов.

Контрольные вопросы

1. Что отражает модель жизненного цикла ИС?

Организационные процессы внедрения ИС

События, происходящие с системой в процессе ее создания и использования
Процесс проектирования ИС

2. Укажите свойства каскадной модели ЖЦ

Предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке

Предусматривает разработку итерациями, с циклами обратной связи между этапами

Переход на следующий этап означает полное завершение работ на предыдущем этапе

Время жизни каждого из этапов растягивается на весь период разработки

3. Укажите свойства спиральной модели ЖЦ

Позволяет планировать сроки завершения всех работ и соответствующие затраты

На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта

Переход на следующий этап означает полное завершение работ на предыдущем этапе

Требования проекта постоянно уточняются

На каждом витке спирали планируются работы следующего витка

4. Укажите свойства поэтапной модели ЖЦ с промежуточным контролем

Время жизни каждого из этапов растягивается на весь период разработки

Учитывает взаимовлияние результатов разработки на различных этапах

На каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности

Переход на следующий этап означает полное завершение работ на предыдущем этапе

5. Какую модель жизненного цикла следует использовать при создании простых ИС?

Поэтапную модель с промежуточным контролем

Спиральную модель

Каскадную модель

6. Какая модель жизненного цикла наиболее объективно отражает реальный процесс создания сложных систем?

Поэтапная модель с промежуточным контролем

Каскадная модель

Спиральная модель

7. Какие из перечисленных процессов относятся к группе основных в соответствии со стандартом ISO/IEC 12207?

Поставка

Обеспечение качества

Верификация

Управление конфигурацией

Документирование

Разработка

Приобретение

Набрано баллов

3 Организация разработки ИС

3.1 Каноническое проектирование ИС

Организация канонического проектирования ИС ориентирована на использование главным образом каскадной модели жизненного цикла ИС. Стадии и этапы работы описаны в стандарте ГОСТ 34.601-90.

В зависимости от сложности объекта автоматизации и набора задач, требующих решения при создании конкретной ИС, стадии и этапы работ могут иметь различную трудоемкость. Допускается объединять последовательные этапы и даже исключать некоторые из них на любой стадии проекта. Допускается также начинать выполнение работ следующей стадии до окончания предыдущей.

Стадии и этапы создания ИС, выполняемые организациями-участниками, прописываются в договорах и технических заданиях на выполнение работ:

Стадия 1. **Формирование требований к ИС.**

На начальной стадии проектирования выделяют следующие этапы работ:

- обследование объекта и обоснование необходимости создания ИС;
- формирование требований пользователей к ИС;
- оформление отчета о выполненной работе и тактико-технического задания на разработку.

Стадия 2. **Разработка концепции ИС.**

- изучение объекта автоматизации;
- проведение необходимых научно-исследовательских работ;
- разработка вариантов концепции ИС, удовлетворяющих требованиям пользователей;
- оформление отчета и утверждение концепции.

Стадия 3. **Техническое задание.**

- разработка и утверждение технического задания на создание ИС.

Стадия 4. **Эскизный проект.**

- разработка предварительных проектных решений по системе и ее частям;
- разработка эскизной документации на ИС и ее части.

Стадия 5. **Технический проект.**

- разработка проектных решений по системе и ее частям;
- разработка документации на ИС и ее части;
- разработка и оформление документации на поставку комплектующих изделий;

- разработка заданий на проектирование в смежных частях проекта.

Стадия 6. **Рабочая документация.**

- разработка рабочей документации на ИС и ее части;
- разработка и адаптация программ.

Стадия 7. **Ввод в действие.**

- подготовка объекта автоматизации;
- подготовка персонала;
- комплектация ИС поставляемыми изделиями (программными и техническими средствами, программными техническими комплексами, информационными изделиями);
- строительно-монтажные работы;
- пусконаладочные работы;
- проведение предварительных испытаний;
- проведение опытной эксплуатации;
- проведение приемочных испытаний.

Стадия 8. **Сопровождение ИС.**

- выполнение работ в соответствии с гарантийными обязательствами;
- послегарантийное обслуживание.

Обследование- это изучение и диагностический анализ организационной структуры предприятия, его деятельности и существующей системы обработки информации. Материалы, полученные в результате обследования, используются для:

- обоснования разработки и поэтапного внедрения систем;
- составления технического задания на разработку систем;
- разработки технического и рабочего проектов систем.

На этапе обследования целесообразно выделить две составляющие: определение стратегии внедрения ИС и детальный анализ деятельности организации.

Основная задача первого этапа обследования - оценка реального объема проекта, его целей и задач на основе выявленных функций и информационных элементов автоматизируемого объекта высокого уровня. Эти задачи могут быть реализованы или заказчиком ИС самостоятельно, или с привлечением консалтинговых организаций. Этап предполагает тесное взаимодействие с основными потенциальными пользователями системы и бизнес-экспертами. Основная задача взаимодействия - получить полное и однозначное понимание требований заказчика. Как правило, нужная информация может быть получена в результате интервью, бесед или семинаров с руководством, экспертами и пользователями.

По завершении этой стадии обследования появляется возможность определить вероятные технические подходы к созданию системы и оценить затраты на ее реализацию (затраты на аппаратное обеспечение, закупаемое программное обеспечение и разработку нового программного обеспечения).

Результатом этапа определения стратегии является документ (технико-экономическое обоснование проекта), где четко сформулировано, что получит заказчик, если согласится финансировать проект, когда он получит готовый продукт (график выполнения работ) и сколько это будет стоить (для крупных проектов должен быть составлен график финансирования на разных этапах работ). В документе желательно отразить не только затраты, но и выгоду проекта, например время окупаемости проекта, ожидаемый экономический эффект (если его удастся оценить).

Ориентировочное содержание этого документа:

- ограничения, риски, критические факторы, которые могут повлиять на успешность проекта;
- совокупность условий, при которых предполагается эксплуатировать будущую систему: архитектура системы, аппаратные и программные ресурсы, условия функционирования, обслуживающий персонал и пользователи системы;
- сроки завершения отдельных этапов, форма приемки/сдачи работ, привлекаемые ресурсы, меры по защите информации;
- описание выполняемых системой функций;
- возможности развития системы;
- информационные объекты системы;
- интерфейсы и распределение функций между человеком и системой;

- требования к программным и информационным компонентам ПО, требования к СУБД;
- что не будет реализовано в рамках проекта.

На этапе детального анализа деятельности организации изучаются задачи, обеспечивающие реализацию функций управления, организационная структура, штаты и содержание работ по управлению предприятием, а также характер подчиненности вышестоящим органам управления. На этом этапе должны быть выявлены:

- инструктивно-методические и директивные материалы, на основании которых определяются состав подсистем и перечень задач;
- возможности применения новых методов решения задач.

Аналитики собирают и фиксируют информацию в двух взаимосвязанных формах:

- функции - информация о событиях и процессах, которые происходят в бизнесе;
- сущности - информация о вещах, имеющих значение для организации и о которых что-то известно.

При изучении каждой функциональной задачи управления определяются:

- наименование задачи; сроки и периодичность ее решения;
- степень формализуемости задачи;
- источники информации, необходимые для решения задачи;
- показатели и их количественные характеристики;

- порядок корректировки информации;
- действующие алгоритмы расчета показателей и возможные методы контроля;
- действующие средства сбора, передачи и обработки информации;
- действующие средства связи;
- принятая точность решения задачи;
- трудоемкость решения задачи;
- действующие формы представления исходных данных и результатов их обработки в виде документов;
- потребители результатной информации по задаче.

Одной из наиболее трудоемких, хотя и хорошо формализуемых задач этого этапа является описание документооборота организации. При обследовании документооборота составляется схема маршрута движения документов, которая должна отразить:

- количество документов;
- место формирования показателей документа;
- взаимосвязь документов при их формировании;
- маршрут и длительность движения документа;
- место использования и хранения данного документа;

- внутренние и внешние информационные связи;
- объем документа в знаках.

По результатам обследования устанавливается перечень задач управления, решение которых целесообразно автоматизировать, и очередность их разработки.

На этапе обследования следует классифицировать планируемые функции системы по степени важности. Один из возможных форматов представления такой классификации - MuSCoW.

Эта аббревиатура расшифровывается так: Must have - необходимые функции; Should have - желательные функции; Could have - возможные функции; Won't have - отсутствующие функции.

Функции первой категории обеспечивают критичные для успешной работы системы возможности.

Реализация функций второй и третьей категорий ограничивается временными и финансовыми рамками: разрабатывается то, что необходимо, а также максимально возможное в порядке приоритета число функций второй и третьей категорий.

Последняя категория функций особенно важна, поскольку необходимо четко представлять границы проекта и набор функций, которые будут отсутствовать в системе.

Модели деятельности организации создаются в двух видах:

- модель "как есть"("as-is")- отражает существующие в организации бизнес-процессы;
- модель "как должно быть"("to-be") - отражает необходимые изменения бизнес-процессов с учетом внедрения ИС.

На этапе анализа необходимо привлекать к работе группы тестирования для решения следующих задач:

- получения сравнительных характеристик предполагаемых к использованию аппаратных платформ, операционных систем, СУБД, иного окружения;

- разработки плана работ по обеспечению надежности информационной системы и ее тестирования.

Привлечение тестировщиков на ранних этапах разработки является целесообразным для любых проектов. Если проектное решение оказалось неудачным и это обнаружено слишком поздно (на этапе разработки или, что еще хуже, на этапе внедрения в эксплуатацию), то исправление ошибки проектирования обходится очень дорого. Чем раньше группы тестирования выявляют ошибки в информационной системе, тем ниже стоимость сопровождения системы. Время на тестирование системы и на исправление обнаруженных ошибок следует предусматривать не только на этапе разработки, но и на этапе проектирования.

Для автоматизации тестирования следует использовать системы отслеживания ошибок (bug tracking). Это позволяет иметь единое хранилище ошибок, отслеживать их повторное появление, контролировать скорость и эффективность исправления ошибок, видеть наиболее нестабильные компоненты системы, а также поддерживать связь между группой разработчиков и группой тестирования (уведомления об изменениях по e-mail и т.п.). Чем больше проект, тем сильнее потребность в bug tracking.

Результаты обследования представляют объективную основу для формирования технического задания на информационную систему.

Техническое задание – это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления.

При разработке технического задания необходимо решить следующие задачи:

- установить общую цель создания ИС, определить состав подсистем и функциональных задач;
- разработать и обосновать требования, предъявляемые к подсистемам;

- разработать и обосновать требования, предъявляемые к информационной базе, математическому и программному обеспечению, комплексу технических средств (включая средства связи и передачи данных);
- установить общие требования к проектируемой системе;
- определить перечень задач создания системы и исполнителей;
- определить этапы создания системы и сроки их выполнения;
- провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения.

3.1.1 Состав и содержание технического задания (ГОСТ 34.602-89)

1. Общие сведения:

- полное наименование системы и ее условное обозначение
- шифр темы или шифр (номер) договора;
- наименование предприятий разработчика и заказчика системы, их реквизиты
- перечень документов, на основании которых создается ИС
- плановые сроки начала и окончания работ
- сведения об источниках и порядке финансирования работ
- порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств

2. Назначение и цели создания (развития) системы:

- вид автоматизируемой деятельности
- перечень объектов, на которых предполагается использование системы
- наименования и требуемые значения технических, технологических, производственно-экономических и др. показателей объекта, которые должны быть достигнуты при внедрении ИС

3. Характеристика объектов автоматизации:

- краткие сведения об объекте автоматизации
- сведения об условиях эксплуатации и характеристиках окружающей среды

4. Требования к системе.

Требования к системе в целом:

- требования к структуре и функционированию системы (перечень подсистем, уровни иерархии, степень централизации, способы информационного обмена, режимы функционирования, взаимодействие со смежными системами, перспективы развития системы)
- требования к персоналу (численность пользователей, квалификация, режим работы, порядок подготовки)
- показатели назначения (степень приспособляемости системы к изменениям процессов управления и значений параметров)
- требования к надежности, безопасности, эргономике, транспортабельности, эксплуатации, техническому обслуживанию и ремонту, защите и сохранности информации, защите от внешних воздействий, к патентной чистоте, по стандартизации и унификации

Требования к функциям (по подсистемам) :

- перечень подлежащих автоматизации задач
- временной регламент реализации каждой функции
- требования к качеству реализации каждой функции, к форме представления выходной информации, характеристики точности, достоверности выдачи результатов
- перечень и критерии отказов

Требования к видам обеспечения:

- математическому (состав и область применения мат. моделей и методов, типовых и разрабатываемых алгоритмов)
- информационному (состав, структура и организация данных, обмен данными между компонентами системы, информационная совместимость со смежными системами, используемые классификаторы, СУБД, контроль данных и ведение информационных массивов, процедуры придания юридической силы выходным документам)
- лингвистическому (языки программирования, языки взаимодействия пользователей с системой, системы кодирования, языки ввода- вывода)
- программному (независимость программных средств от платформы, качество программных средств и способы его контроля, использование фондов алгоритмов и программ)
- техническому
- метрологическому
- организационному (структура и функции эксплуатирующих подразделений, защита от ошибочных действий персонала)

- методическому (состав нормативно- технической документации)

5. Состав и содержание работ по созданию системы:

- перечень стадий и этапов работ
- сроки исполнения
- состав организаций — исполнителей работ
- вид и порядок экспертизы технической документации
- программа обеспечения надежности
- программа метрологического обеспечения

6. Порядок контроля и приемки системы:

- виды, состав, объем и методы испытаний системы
- общие требования к приемке работ по стадиям
- статус приемной комиссии

7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие:

- преобразование входной информации к машиночитаемому виду
- изменения в объекте автоматизации
- сроки и порядок комплектования и обучения персонала

8. Требования к документированию:

- перечень подлежащих разработке документов
- перечень документов на машинных носителях

9. Источники разработки: документы и информационные материалы, на основании которых разрабатывается ТЗ и система.

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям.

Выполнение стадии эскизного проектирования не является строго обязательной. Если основные проектные решения определены ранее или достаточно очевидны для конкретной ИС и объекта автоматизации, то эта стадия может быть исключена из общей последовательности работ.

Содержание эскизного проекта задается в ТЗ на систему. Как правило, на этапе эскизного проектирования определяются:

- функции ИС;
- функции подсистем, их цели и ожидаемый эффект от внедрения;
- состав комплексов задач и отдельных задач;
- концепция информационной базы и ее укрупненная структура;
- функции системы управления базой данных;
- состав вычислительной системы и других технических средств;
- функции и параметры основных программных средств.

По результатам проделанной работы оформляется, согласовывается и утверждается документация в объеме, необходимом для описания полной совокупности принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию системы.

На основе технического задания (и эскизного проекта) разрабатывается технический проект ИС. Технический проект системы - это техническая документация, содержащая общесистемные проектные решения, алгоритмы решения задач, а также оценку экономической эффективности автоматизированной системы управления и перечень мероприятий по подготовке объекта к внедрению.

На этом этапе осуществляется комплекс научно-исследовательских и экспериментальных работ для выбора основных проектных решений и расчет экономической эффективности системы.

3.1.2 Содержание технического проекта (ГОСТ 34.602-89)

1. Пояснительная записка:

- основания для разработки системы
- перечень организаций разработчиков
- краткая характеристика объекта с указанием основных технико-экономических показателей его функционирования и связей с другими объектами
- краткие сведения об основных проектных решениях по функциональной и обеспечивающим частям системы

2. Функциональная и организационная структура системы:

- обоснование выделяемых подсистем, их перечень и назначение
- перечень задач, решаемых в каждой подсистеме, с краткой характеристикой их содержания

- схема информационных связей между подсистемами и между задачами в рамках каждой подсистемы

3. Постановка задач и алгоритмы решения:

- организационно-экономическая сущность задачи (наименование, цель решения, краткое содержание, метод, периодичность и время решения задачи, способы сбора и передачи данных, связь задачи с другими задачами, характер использования результатов решения, в которых они используются)
- экономико-математическая модель задачи (структурная и развернутая форма представления)
- входная оперативная информация (характеристика показателей, диапазон изменения, формы представления)
- нормативно-справочная информация (НСИ) (содержание и формы представления)
- информация, хранимая для связи с другими задачами
- информация, накапливаемая для последующих решений данной задачи
- информация по внесению изменений (система внесения изменений и перечень информации, подвергающейся изменениям)
- алгоритм решения задачи (последовательность этапов расчета, схема, расчетные формулы)
- контрольный пример (набор заполненных данными форм входных документов, условные документы с накапливаемой и хранимой информацией, формы выходных документов, заполненные по результатам решения экономико-технической задачи и в соответствии с разработанным алгоритмом расчета)

4. Организация информационной базы:

- источники поступления информации и способы ее передачи
- совокупность показателей, используемых в системе
- состав документов, сроки и периодичность их поступления
- основные проектные решения по организации фонда НСИ
- состав НСИ, включая перечень реквизитов, их определение, диапазон изменения и перечень документов НСИ
- перечень массивов НСИ, их объем, порядок и частота корректировки информации
- структура фонда НСИ с описанием связи между его элементами; требования к технологии создания и ведения фонда
- методы хранения, поиска, внесения изменений и контроля
- определение объемов и потоков информации НСИ
- контрольный пример по внесению изменений в НСИ
- предложения по унификации документации

5. Альбом форм документов

6. Система математического обеспечения:

- обоснование структуры математического обеспечения
- обоснование выбора системы программирования
- перечень стандартных программ

7. Принцип построения комплекса технических средств:

- описание и обоснование схемы технологического процесса обработки данных
- обоснование и выбор структуры комплекса технических средств и его функциональных групп
- обоснование требований к разработке нестандартного оборудования
- комплекс мероприятий по обеспечению надежности функционирования технических средств

8. Расчет экономической эффективности системы:

- сводная смета затрат, связанных с эксплуатацией систем
- расчет годовой экономической эффективности, источниками которой являются оптимизация производственной структуры хозяйства (объединения), снижение себестоимости продукции за счет рационального использования производственных ресурсов и уменьшения потерь, улучшения принимаемых управленческих решений

9. Мероприятия по подготовке объекта к внедрению системы:

- перечень организационных мероприятий по совершенствованию бизнес-процессов
- перечень работ по внедрению системы, которые необходимо выполнить на стадии рабочего проектирования, с указанием сроков и ответственных лиц

10. Ведомость документов

В завершение стадии технического проектирования производится разработка документации на поставку серийно выпускаемых изделий для комплектования ИС, а также определяются технические требования и составляются ТЗ на разработку изделий, не изготавливаемых серийно.

На стадии "рабочая документация" осуществляется создание программного продукта и разработка всей сопровождающей документации. Документация должна содержать все необходимые и достаточные сведения для обеспечения выполнения работ по вводу ИС в действие и ее эксплуатации, а также для поддержания уровня эксплуатационных характеристик (качества) системы. Разработанная документация должна быть соответствующим образом оформлена, согласована и утверждена.

Для ИС, которые являются разновидностью автоматизированных систем, устанавливают следующие основные виды испытаний: предварительные, опытная эксплуатация и приемочные. При необходимости допускается дополнительно проведение других видов испытаний системы и ее частей.

В зависимости от взаимосвязей частей ИС и объекта автоматизации испытания могут быть автономные или комплексные. Автономные испытания охватывают части системы. Их проводят по мере готовности частей системы к сдаче в опытную эксплуатацию. Комплексные испытания проводят для групп взаимосвязанных частей или для системы в целом.

Для планирования проведения всех видов испытаний разрабатывается документ "Программа и методика испытаний". Разработчик документа устанавливается в договоре или ТЗ. В качестве приложения в документ могут включаться тесты или контрольные примеры.

Предварительные испытания проводят для определения работоспособности системы и решения вопроса о возможности ее приемки в опытную эксплуатацию. Предварительные испытания следует выполнять после проведения разработчиком отладки и тестирования поставляемых программных и технических средств системы и представления им соответствующих документов об их готовности к испытаниям, а также после ознакомления персонала ИС с эксплуатационной документацией.

Опытную эксплуатацию системы проводят с целью определения фактических значений количественных и качественных характеристик системы и готовности персонала к работе в условиях ее функционирования, а также определения фактической эффективности и корректировки, при необходимости, документации.

Приемочные испытания проводят для определения соответствия системы техническому заданию,

оценки качества опытной эксплуатации и решения вопроса о возможности приемки системы в постоянную эксплуатацию.

3.2 Типовое проектирование ИС

Методы типового проектирования ИС достаточно подробно рассмотрены в литературе. В данной книге приведены основные определения и представлено задание для разработки проекта ИС методом типового проектирования (кейс "Проектирование ИС предприятия оптовой торговли лекарственными препаратами").

Типовое проектирование ИС предполагает создание системы из готовых типовых элементов. Основопологающим требованием для применения методов типового проектирования является возможность декомпозиции проектируемой ИС на множество составляющих компонентов (подсистем, комплексов задач, программных модулей и т.д.). Для реализации выделенных компонентов выбираются имеющиеся на рынке типовые проектные решения, которые настраиваются на особенности конкретного предприятия.

Типовое проектное решение (ТПР)- это тиражируемое (пригодное к многократному использованию) проектное решение.

Принятая классификация ТПР основана на уровне декомпозиции системы. Выделяются следующие классы ТПР:

- элементные ТПР - типовые решения по задаче или по отдельному виду обеспечения задачи (информационному, программному, техническому, математическому, организационному);
- подсистемные ТПР - в качестве элементов типизации выступают отдельные подсистемы, разработанные с учетом функциональной полноты и минимизации внешних информационных связей;

- объектные ТПР - типовые отраслевые проекты, которые включают полный набор функциональных и обеспечивающих подсистем ИС.

Каждое типовое решение предполагает наличие, кроме собственно функциональных элементов (программных или аппаратных), документации с детальным описанием ТПР и процедур настройки в соответствии с требованиями разрабатываемой системы.

Выделим достоинства и недостатки ТПР.

Элементные ТПР Библиотеки методо-ориентированных программ:

- обеспечивается применение модульного подхода к проектированию и документированию ИС
- большие затраты времени на сопряжение разнородных элементов вследствие информационной, программной и технической несовместимости
- большие затраты времени на доработку ТПР отдельных элементов

Подсистемные ТПР Пакеты прикладных программ:

- достигается высокая степень интеграции элементов ИС
- позволяют осуществлять: модульное проектирование; параметрическую настройку программных компонентов на различные объекты управления
- обеспечивают: сокращение затрат на проектирование и программирование взаимосвязанных компонентов; хорошее документирование отображаемых процессов обработки информации
- адаптивность ТПР недостаточна с позиции непрерывного инжиниринга деловых процессов

- возникают проблемы в комплексировании разных функциональных подсистем, особенно в случае использования решений нескольких производителей программного обеспечения

Объектные ТПР Отраслевые проекты ИС:

- комплексирование всех компонентов ИС за счет методологического единства и информационной, программной и технической совместимости
- открытость архитектуры — позволяет устанавливать ТПР на разных программно-технических платформах
- масштабируемость — допускает конфигурацию ИС для переменного числа рабочих мест
- конфигурируемость — позволяет выбирать необходимое подмножество компонентов
- проблемы привязки типового проекта к конкретному объекту управления, что вызывает в некоторых случаях даже необходимость изменения организационно-экономической структуры объекта автоматизации

Для реализации типового проектирования используются два подхода: параметрически-ориентированное и модельно-ориентированное проектирование.

Параметрически-ориентированное проектирование включает следующие этапы: определение критериев оценки пригодности пакетов прикладных программ (ППП) для решения поставленных задач, анализ и оценка доступных ППП по сформулированным критериям, выбор и закупка наиболее подходящего пакета, настройка параметров (доработка) закупленного ППП.

Критерии оценки ППП делятся на следующие группы:

- назначение и возможности пакета;

- отличительные признаки и свойства пакета;
- требования к техническим и программным средствам;
- документация пакета;
- факторы финансового порядка;
- особенности установки пакета;
- особенности эксплуатации пакета;
- помощь поставщика по внедрению и поддержанию пакета;
- оценка качества пакета и опыт его использования;
- перспективы развития пакета.

Внутри каждой группы критериев выделяется некоторое подмножество частных показателей, детализирующих каждый из десяти выделенных аспектов анализа выбираемых ППП.

Числовые значения показателей для конкретных ППП устанавливаются экспертами по выбранной шкале оценок (например, 10-балльной). На их основе формируются групповые оценки и комплексная оценка пакета (путем вычисления средневзвешенных значений). Нормированные взвешивающие коэффициенты также получают экспертным путем.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации.

Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретного предприятия.

Типовая ИС в специальной базе метаинформации - репозитории - содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование программного обеспечения. Таким образом, модельно-ориентированное проектирование ИС предполагает, прежде всего, построение модели объекта автоматизации с использованием специального программного инструментария (например, SAP Business Engineering Workbench (BEW), BAAN Enterprise Modeler). Возможно также создание системы на базе типовой модели ИС из репозитория, который поставляется вместе с программным продуктом и расширяется по мере накопления опыта проектирования информационных систем для различных отраслей и типов производства.

Репозиторий содержит базовую (ссылочную) модель ИС, типовые (референтные) модели определенных классов ИС, модели конкретных ИС предприятий.

Базовая модель ИС в репозитории содержит описание бизнес-функций, бизнес-процессов, бизнес-объектов, бизнес-правил, организационной структуры, которые поддерживаются программными модулями типовой ИС.

Типовые модели описывают конфигурации информационной системы для определенных отраслей или типов производства.

Модель конкретного предприятия строится либо путем выбора фрагментов основной или типовой модели в соответствии со специфическими особенностями предприятия (BAAN Enterprise Modeler), либо путем автоматизированной адаптации этих моделей в результате экспертного опроса (SAP Business Engineering Workbench).

Построенная модель предприятия в виде метаописания хранится в репозитории и при необходимости может быть откорректирована. На основе этой модели автоматически осуществляется конфигурирование и настройка информационной системы.

Бизнес-правила определяют условия корректности совместного применения различных компонентов ИС и используются для поддержания целостности создаваемой системы.

Модель бизнес-функций представляет собой иерархическую декомпозицию функциональной дея-

тельности предприятия (подробное описание см. в разделе "Анализ и моделирование функциональной области внедрения ИС").

Модель бизнес-процессов отражает выполнение работ для функций самого нижнего уровня модели бизнес-функций (подробное описание см. в разделе "Спецификация функциональных требований к ИС"). Для отображения процессов используется модель управления событиями (EPC - Event-driven Process Chain). Именно модель бизнес-процессов позволяет выполнить настройку программных модулей - приложений информационной системы в соответствии с характерными особенностями конкретного предприятия.

Модели бизнес-объектов используются для интеграции приложений, поддерживающих исполнение различных бизнес-процессов (подробное описание см. в разделе "Этапы проектирования ИС с применением UML").

Модель организационной структуры предприятия представляет собой традиционную иерархическую структуру подчинения подразделений и персонала (подробное описание см. в разделе "Анализ и моделирование функциональной области внедрения ИС").

Внедрение типовой информационной системы начинается с анализа требований к конкретной ИС, которые выявляются на основе результатов предпроектного обследования объекта автоматизации (см. раздел "Анализ и моделирование функциональной области внедрения ИС"). Для оценки соответствия этим требованиям программных продуктов может использоваться описанная выше методика оценки ППП. После выбора программного продукта на базе имеющихся в нем референтных моделей строится предварительная модель ИС, в которой отражаются все особенности реализации ИС для конкретного предприятия. Предварительная модель является основой для выбора типовой модели системы и определения перечня компонентов, которые будут реализованы с использованием других программных средств или потребуют разработки с помощью имеющихся в составе типовой ИС инструментальных средств (например, АВАР в SAP, Tools в BAAN).

Реализация типового проекта предусматривает выполнение следующих операций:

- установку глобальных параметров системы;
- задание структуры объекта автоматизации;
- определение структуры основных данных;
- задание перечня реализуемых функций и процессов;
- описание интерфейсов;
- описание отчетов;
- настройку авторизации доступа;
- настройку системы архивирования.

3.3 ISO/IEC 12207

В соответствии с базовым международным стандартом ISO/IEC 12207 все процессы ЖЦ ПО делятся на три группы:

1. Основные процессы:

- приобретение;
- поставка;
- разработка;

- эксплуатация;
- сопровождение.

2. Вспомогательные процессы:

- документирование;
- управление конфигурацией;
- обеспечение качества;
- разрешение проблем;
- аудит;
- аттестация;
- совместная оценка;
- верификация.

3. Организационные процессы:

- создание инфраструктуры;
- управление;
- обучение;
- усовершенствование.

В таблице 3.1 приведены ориентировочные описания основных процессов ЖЦ. Вспомогательные процессы предназначены для поддержки выполнения основных процессов, обеспечения качества проекта, организации верификации, проверки и тестирования ПО. Организационные процессы определяют действия и задачи, выполняемые как заказчиком, так и разработчиком проекта для управления своими процессами.

Позднее был разработан и в 2002 г. опубликован стандарт на процессы жизненного цикла систем (ISO/IEC 15288 System life cycle processes). К разработке стандарта были привлечены специалисты различных областей: системной инженерии, программирования, управления качеством, человеческими ресурсами, безопасностью и пр. Был учтен практический опыт создания систем в государственных, коммерческих, военных и академических организациях. Стандарт применим для широкого класса систем, но его основное предназначение - поддержка создания компьютеризированных систем.

Таблица 3.1: Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

| Действия | Вход | Результат |
|---|---|--|
| Приобретение (заказчик) | | |
| Инициирование. Подготовка заявочных предложений. Подготовка договора. Контроль деятельности поставщика. Приемка ИС. | Решение о начале работ по внедрению ИС. Результаты обследования деятельности заказчика. Результаты анализа рынка ИС/ тендера. План поставки/ разработки. Комплексный тест ИС. | Технико-экономическое обоснование внедрения ИС. Техническое задание на ИС. Договор на поставку/ разработку. Акты приемки этапов работы. Акт приемно-сдаточных испытаний. |
| Поставка (разработчик ИС) | | |
| Инициирование. Ответ на заявочные предложения. Подготовка договора. Планирование исполнения. Поставка ИС. | Техническое задание на ИС. Решение руководства об участии в разработке. Результаты тендера. Техническое задание на ИС. План управления проектом. Разработанная ИС и документация. | Решение об участии в разработке. Коммерческие предложения/ конкурсная заявка. Договор на поставку/ разработку. План управления проектом. Реализация/ корректировка. Акт приемно-сдаточных испытаний. |

| Действия | Вход | Результат |
|---|--|---|
| Разработка (разработчик ИС) | | |
| Подготовка. Анализ требований к ИС. Проектирование архитектуры ИС. Разработка требований к ПО. Проектирование архитектуры ПО. Детальное проектирование ПО. Кодирование и тестирование ПО. Интеграция ПО и квалификационное тестирование ПО. Интеграция ИС и квалификационное тестирование ИС. | Техническое задание на ИС. Техническое задание на ИС, модель ЖЦ. Техническое задание на ИС. Подсистемы ИС. Спецификации требования к компонентам ПО. Архитектура ПО. Материалы детального проектирования ПО. План интеграции ПО, тесты. Архитектура ИС, ПО, документация на ИС, тесты. | Используемая модель ЖЦ, стандарты разработки. План работ. Состав подсистем, компоненты оборудования. Спецификации требования к компонентам ПО. Состав компонентов ПО, интерфейсы с БД, план интеграции ПО. Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам. Тексты модулей ПО, акты автономного тестирования. Оценка соответствия комплекса ПО требованиям ТЗ. Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ. |

Согласно стандарту ISO/IEC серии 15288 в структуру ЖЦ следует включать следующие группы процессов:

1. Договорные процессы:

- приобретение (внутренние решения или решения внешнего поставщика);
- поставка (внутренние решения или решения внешнего поставщика).

2. Процессы предприятия:

- управление окружающей средой предприятия;

- инвестиционное управление;
- управление ЖЦ ИС;
- управление ресурсами;
- управление качеством.

3. Проектные процессы:

- планирование проекта;
- оценка проекта;
- контроль проекта;
- управление рисками;
- управление конфигурацией;
- управление информационными потоками;
- принятие решений.

4. Технические процессы:

- определение требований;
- анализ требований;
- разработка архитектуры;
- внедрение;
- интеграция;

- верификация;
- переход;
- аттестация;
- эксплуатация;
- сопровождение;
- утилизация.

5. Специальные процессы:

- определение и установка взаимосвязей исходя из задач и целей.

Стадии создания системы, предусмотренные в стандарте ISO/IEC 15288, несколько отличаются от рассмотренных выше. Перечень стадий и основные результаты, которые должны быть достигнуты к моменту их завершения:

1. **Формирование концепции.** Анализ потребностей, выбор концепции и проектных решений.
2. **Разработка.** Проектирование системы.
3. **Реализация.** Изготовление системы.
4. **Эксплуатация.** Ввод в эксплуатацию и использование системы.
5. **Поддержка.** Обеспечение функционирования системы.
6. **Снятие с эксплуатации.** Прекращение использования, демонтаж, архивирование.

3.4 Экстремальное программирование

3.4.1 Теория

Экстремальное программирование (ХР) – методология быстрой разработки программного обеспечения. Состоит из набора методик и принципов, позволяющих как по отдельности, так и в комплексе, оптимизировать процесс разработки. Этот подход также регламентирует права разработчиков и заказчиков.

Права и роли

В экстремальном программировании чётко описаны все роли. Каждая роль предусматривает характерный набор прав и обязанностей. Здесь существуют две ключевые роли: заказчик и разработчик.

Заказчик

Человек или группа людей, заинтересованных в создании конкретного программного продукта. Он имеет следующие права и обязанности:

- зафиксировать сроки выпуска версий продукта;
- принимать решения относительно запланированных составляющих программы;
- знать ориентировочную стоимость каждой функциональной составляющей;
- принимать важные бизнес - решения;
- знать текущее состояние системы;

- изменять требования к системе, когда это действительно важно.

Для успешного использования своих прав заказчик должен полагаться на данные, предоставляемые разработчиками.

Разработчик

Один или группа от двух до десяти человек, занимающихся непосредственно программированием и сопутствующими инженерными вопросами. Разработчик наделён следующими правами и обязанностями:

- получить достаточное знание вопросов, которые должны быть запрограммированы;
- иметь возможность выяснения деталей в процессе разработки;
- предоставлять ориентировочные, но откровенные оценки трудозатрат на каждую функциональную часть или историю пользователя;
- корректировать оценки в пользу более точных в процессе разработки;
- предоставлять оценку рисков, связанных с использованием конкретных технологий.

Роли внутри роли

Каждая из базовых ролей экстремального программирования может быть уточнена более мелкими ролями. В XP разрешено совмещение ролей в рамках одного человека.

Сторона заказчика

Составитель историй – специалист предметной области, обладающий способностями доступно изложить и описать требования к разрабатываемой системе. Этот человек или группа людей ответственны за написание историй пользователя и прояснения недопонимания со стороны программистов.

Приёмщик – человек, контролирующий правильность функционирования системы. Хорошо владеет предметной областью. В обязанности входит написание приёмочных тестов.

Большой босс – следит за работой всех звеньев, от разработчиков до конечных пользователей. Он контролирует внедрение системы и сопутствующие организационные моменты. Может быть также инвестором проекта.

Сторона разработчика

Программист – человек, занимающийся кодированием и проектированием на низком уровне. Он достаточно компетентен для решения текущих задач разработки и предоставления правдивых оценок запланированным задачам.

Инструктор – опытный разработчик, хорошо владеющий всем процессом разработки и его методиками. Несёт ответственность за обучение команды аспектам процесса разработки. Внедряет и контролирует правильность выполнения методик используемого процесса. Обращает внимание команды на важные, но по каким-то причинам упущенные моменты разработки. Вместе с тем инструктор, как и любой другой человек, не всезнающий и с вниманием относится к идеям других членов команды.

Наблюдатель – член команды разработчиков, пользующийся доверием всей группы, который следит за прогрессом разработки. Он сравнивает предварительные оценки трудозатрат и реально потраченные, выводя количественные показатели работы команды. Это такие как средняя скорость и процентное соотношение выполненных и запланированных задач. Данная информация предоставляется заказчику для своевременного контроля над ситуацией. Часть этой информации ненавязчиво

предоставляется и разработчикам, для знания состояния проекта, мест возникновения затруднений и что ещё можно успеть сделать.

Дипломат – коммуникабельная личность, инициирующая общение между членами команды. Так как документооборот минимизирован, важно постоянное общение и передача опыта внутри команды, лучшее понимание требований к системе. Дипломат регулирует и упрощает общение между заказчиками и разработчиками. Является важным звеном на собраниях. Он препятствует недомолвкам, разгару страстей и ненужным ссорам. Дипломат не может навязывать своего мнения дискуссирующим.

Внешние роли

Консультант – специалист, обладающий конкретными техническими навыками, для помощи разработчикам в трудно разрешимых задачах. Обычно привлекается со стороны.

3.4.2 Правила

Соглашение о кодировании

Вы в команде, которая работает над данным проектом продолжительное время. Люди приходят и уходят. Никто не кодирует в одиночку и код принадлежит всем. Всегда будут моменты, когда необходимо будет понять и скорректировать чужой код. Разработчики будут удалять или изменять дублирующий код, анализировать и улучшать чужие классы и т.п. Со временем нельзя будет сказать кто автор конкретного класса.

Следовательно, все должны подчиняться общим стандартам кодирования - форматирование кода, именование классов, переменных, констант, стиль комментариев. Таким образом, мы будем уверены,

что внося изменения в чужой код (что необходимо для агрессивного и экстремального продвижения вперед) мы не превратим его в Вавилонское Столпотворение.

Вышесказанное означает, что все члены команды должны договориться об общих стандартах кодирования. Неважно каких. Правило заключается в том, что все им подчиняются. Те, кто не желает их соблюдать покидает команду.

Коллективное владение кодом

Коллективное владение кодом стимулирует разработчиков подавать идеи для всех частей проекта, а не только для своих модулей. Любой разработчик может изменять любой код для расширения функциональности, исправления ошибок или рефакторинга.

С первого взгляда это выглядит как хаос. Однако принимая во внимание что как минимум любой код создан парой разработчиков, что Unit тесты позволяют проверить корректность внесенных изменений и что в реальной жизни все равно так или иначе приходится разбираться в чужом коде, становится ясно что коллективное владение кодом значительно упрощает внесение изменений и снижает риск связанный с высокой специализацией того или иного члена команды.

CRC Сессия

Используйте Class, Responsibilities, Collaboration (CRC - Класс, Обязанности, Взаимодействие) карточки для дизайна системы командой. Использование карточек позволяет легче приучиться мыслить объектами а не функциями и процедурами. Также карточки позволяют большему количеству людей участвовать в процессе дизайна (в идеале - всей команде), а чем больше людей делает дизайн, тем больше интересных идей будет привнесено.

Каждая CRC карточка представляет собой экземпляр объекта. Класс объекта может быть написан сверху, обязанности слева, взаимодействия справа. Мы говорим "могут быть написаны поскольку

когда CRC сессия в разгаре, участникам обычно нужно небольшое число карточек с именами классов и не обязательно они должны быть полностью заполнены.

CRC сессия происходит так: каждый из участников воспроизводит систему говоря какие сообщения он шлет каким объектам. Проходит по всему процессу сообщение за сообщением. Слабые места и проблемы сразу выявляются. Альтернативы дизайна также хорошо видны при симуляции процесса. Для наведения порядка часто используется ограничение числа одновременно взаимодействующих двумя.

Заказчик

Одно из требований XP - заказчик всегда доступен. Он должен не просто помогать команде разработчиков, а быть ее членом. Все фазы XP проекта требуют коммуникации с заказчиком, лучше всего лицом к лицу - на месте. Лучше всего, просто назначить одного или нескольких заказчиков в команду разработчиков. Остерегайтесь, заказчик потребуется на длительное время, и контора заказчика может постараться отдать вам какого-нибудь стажера в качестве эксперта. Вам нужен эксперт.

User Stories пишутся заказчиком с помощью разработчиков. Заказчик помогает удостовериться что большинство желаемых функций системы покрыто User Story.

Во время планирования релиза заказчику необходимо обсудить выбор User Stories которые будут включены в планируемый релиз. Также, возможно, потребуется согласовывать период самого релиза. Заказчик принимает решения относящиеся к целям бизнеса.

Выбирайте самое простое решение

Простой дизайн всегда легче реализовать, чем сложный. Поэтому всегда делайте простейшее решение, которое может работать. Если находите что-нибудь сложное - замените это чем-нибудь простым.

Всегда оказывается быстрее и дешевле заменить сложный код простым до того как начнешь разбираться в сложном коде. Рефакторите чужой код если он кажется вам сложным. Если что-то выглядит сложным - это верный признак проблемы в коде.

Сохраняйте решения насколько возможно простыми как можно дольше. Никогда не добавляйте функциональность на будущее - до того как появляется в ней необходимость. Однако имейте в виду: сохранять дизайн простым - тяжелая работа.

Функциональные тесты

Приемочные тесты (ранее их также называли Функциональные) пишутся на основе User Story. Они рассматривают систему как черный ящик. Заказчик ответственен за проверку корректности функциональных тестов. Эти тесты используются для проверки работоспособности системы перед выпуском ее в производство. Функциональные тесты автоматизируются так, чтобы имелась возможность их часто запускать. Результат сообщается команде и команда отвечает за планирование исправлений функциональных тестов.

Частая интеграция

Разработчики, по-возможности, должны интегрировать и выпускать свой код каждые несколько часов. В любом случае никогда нельзя держать изменения дольше одного дня. Частая интеграция позволяет избежать отчуждения и фрагментирования в разработке, когда разработчики не могут общаться в смысле обмена идеями или повторного использования кода. Каждый должен работать с самой последней версией.

Каждая пара разработчиков должна отдавать свой код как только для этого появляется разумная возможность. Это может быть когда все UnitTest-ы проходят на 100%. Отдавая изменения несколько

раз в день, Вы сводите проблемы интеграции практически к нулю. Интеграция - это деятельность вида "заплати сейчас или заплати больше позднее". Поэтому интегрируя изменения ежедневно маленькими порциями вы не окажетесь перед необходимостью тратить неделю чтобы связать систему в одно целое непосредственно перед сдачей проекта. Всегда работайте над последней версией системы.

Для менеджера. Если разработчик не отдает изменений дольше одного дня - это ясный индикатор серьезной проблемы. Вы должны немедленно разобраться в чем дело. Весь опыт XP команд говорит что всегда причиной задержки является плохой дизайн и его всегда потом приходится переделывать.

Планирование Итерации

Iteration Planning Meeting созывается перед началом каждой итерации для планирования задач которые будут сделаны в этой итерации. Для итерации выбираются User Stories, которые выбрал заказчик в плане релиза начиная с самых важных для заказчика и самых плохих (сопряженных с риском) для разработчиков. Также в итерацию включаются неработающие Функциональные тесты.

User Stories и невыполненные Функциональные тесты разбиваются на задачи. Задачи записываются на карточках. Эти карточки и есть детальный план на итерацию. Каждая задача должна быть продолжительностью от 1 до 3 идеальных дней. Разработчики разбирают задачи и оценивают продолжительность времени, необходимого для их выполнения. Таким образом, каждый разработчик оценивает сколько времени задача займет именно у него. Это важно, чтобы конечную оценку объема работ делал сам разработчик.

Скорость проекта определяет помещаются ли ваши задачи в итерацию или нет. Общая продолжительность задач запланированных на итерацию не должна превышать скорости, достигнутой в предыдущей итерации. Если вы набрали слишком много, то заказчик должен решить какие User Stories, отложить на следующую итерацию. Если набрали слишком мало, то надо добавить следующую User Story. В некоторых случаях можно попросить заказчика разделить одну из User Story, на

две, чтобы включить часть в текущую итерацию.

Откладывание User Story на следующую итерацию может выглядеть страшно, но не позволяйте себе жертвовать рефакторингом и Unit Test-ами чтобы сделать больше. Задолженность по этим категориям быстро замедлит вашу скорость. Не делайте того, что по-вашему понадобится в следующих итерациях - делайте только то что необходимо для выполнения текущих User Stories.

Следите за скоростью проекта и отложенными User Stories. Вполне возможно, что план релиза придется переделывать каждые три-пять итераций. Это нормально. Ведь план релиза - это взгляд в будущее и естественно ожидать что ваши предсказания придется корректировать.

Итерации

Итеративная разработка увеличивает гибкость процесса. Разделите ваш план на итерации продолжительностью от 2 до 3 недель. Сохраняйте постоянную продолжительность итерации на время проекта. Пусть итерации будут пульсом вашего проекта. Это тот ритм который позволит сделать измерение прогресса и планирование простым и надежным.

Не планируйте задач заранее. Вместо этого собирайте Планирование Итерации в начале каждой итерации чтобы запланировать что будет сделано. Также нарушением правил считается забегать вперед и делать то, что не запланировано в этой итерации. Таким образом, становится возможным держать под контролем изменяющиеся требования Заказчика.

Принимайте всерьез сроки завершения итерации. Измеряйте прогресс в процессе работы. Если видно, что вы не сможете сделать все запланированные задачи к сроку, то снова собирайте Планирование Итерации и оцените задачи заново и отложите часть задач.

Сконцентрируйте усилия на завершении самых важных задач, выбранных Заказчиком, вместо того чтобы иметь несколько незаконченных задач, выбранных разработчиком.

Меняйтесь задачами

Необходимо периодически менять задачи у разработчиков для уменьшения риска концентрации знаний и узких мест в коде. Если только один человек в вашей команде может работать в данной области и этот человек уходит или вам просто надо много всего сделать в данном сегменте программы, вы обнаружите что ваш проект еле-еле продвигается вперед.

Cross Training обычно является важным аспектом в компаниях которые стараются избежать концентрации знаний в одном человеке. Перемещение людей по коду в комбинации с парным программированием незаметно делает Cross Training для вас. Вместо одного человека, который знает все о данном куске кода, каждый в вашей команде знает много о коде в каждом модуле.

Команда становится намного более гибкой если все знают достаточно о каждой части системы чтобы начать работать над ней. Вместо того чтобы ждать пока "гуру" закончит работу над критическим куском кода, несколько программистов могут работать над ним одновременно.

При начале новой итерации каждый разработчик должен перейти на новую задачу. Парное программирование помогает преодолеть проблему адаптации (это значит что новый разработчик может начать работать в паре с опытным в данной области разработчиком).

Такая практика также стимулирует появление новых идей и улучшение кода.

Оставляйте оптимизацию на потом

Никогда не оптимизируйте ничего до окончания кодирования. Никогда не пытайтесь угадать где будут узкие места по производительности. Измеряйте!

Сделайте чтобы это работало, затем чтобы работало правильно, затем чтобы работало быстро.

Парное программирование

Весь код для продукционной системы (а это значит за исключением пробных решений) пишется парами. Два разработчика сидят рядом. Один набирает, другой смотрит. Время от времени они меняются. Не разрешается работать в одиночку. Если по какой-то причине второй из пары пропустил что-то (болел, отходил и т.п.) он обязан просмотреть все изменения сделанные первым.

Звучит необычно, но ХР утверждает что после небольшого периода адаптации большинство людей прекрасно работают в парах. Им даже нравится, поскольку работа делается заметно быстрее. Действует принцип "Одна голова хорошо, а две лучше". Пары обычно находят более оптимальные решения. Кроме того существенно увеличивается качество кода, снижается число ошибок и ускоряется обмен знаниями между разработчиками.

Рефакторинг

Мы, программисты, склонны держаться за дизайн долго после того как он становится неуклюжим. Мы продолжаем повторно использовать неудобный в сопровождении код поскольку он все еще как-то работает и мы боимся испортить его. Но действительно ли это выгодно? ХР принимает точку зрения что это невыгодно. Когда мы убираем избыточность, улучшаем устаревший дизайн убираем неиспользуемые куски - мы делаем рефакторинг. Рефакторинг в конечном итоге экономит время и улучшает качество продукта.

Безжалостно пересматривайте любой код для того, чтобы сохранять дизайн простым по мере разработки. Сохраняйте код ясным и понятным чтобы его было легко понять, модифицировать и расширять. Удостоверьтесь что все написано один и только один раз. В конечном итоге, это занимает меньше времени чем доводить до ума запутанную систему.

План Релиза

План Релиза разрабатывается на собрании по планированию Релиза. Релиз Планы описывают взгляд на весь проект и используются в дальнейшем для планирования итераций.

Важно, чтобы технические люди делали технические решения и люди бизнеса - бизнес решения. Планирование Релиза определяет набор правил, которые позволяют всем принимать свои решения. Эти правила определяют метод выработки удовлетворяющего всех плана работ.

Сущность собрания по планированию релиза для команды разработчиков в том, чтобы оценить каждую User Story в идеальных неделях. Идеальная неделя - это сколько по-вашему займет время выполнение задачи если ничто больше вас не будет отвлекать. Ни зависимостей, ни дополнительных задач, но включая тесты. Заказчик затем решает какие задачи наиболее важны или имеют более высокий приоритет.

User Stories записываются на карточках. Разработчики и Заказчик вместе тасуют карточки на столе пока не получится набор User Stories которые вместе будут составлять первый (или следующий) Релиз. Всем хочется выпустить как можно раньше полезную систему которую можно протестировать.

Релиз можно планировать по времени или по объему. Для того чтобы определить сколько User Stories могут быть реализованы к конкретной дате или сколько реального времени займет данный набор задач используют скорость проекта. Если планируете по времени, умножьте количество итераций на скорость проекта для того чтобы узнать сколько User Story может быть реализовано. При планировании по объему, разделите общее количество идеальных недель необходимых для всех User Stories на скорость проекта и вы получите количество итераций необходимых для окончания релиза.

Если менеджмент не устраивают сроки завершения, может появиться соблазн уменьшить оценки объема работ. Вы никогда не должны этого делать. Заниженные оценки обязательно создадут множество проблем позже. Вместо этого ведите переговоры с менеджерами, разработчиками и за-

казчиками пока не выработаете приемлемый для всех план релиза.

Частые Релизы

Разработчики должны выпускать версии системы пользователям (или бета-тестерам) как можно чаще.

Планирование Релиза используется для поиска небольших функциональных фрагментов имеющих значимый бизнес-смысл и которые могут быть выпущены в реальное окружение на ранних стадиях разработки. Это критично для своевременного получения полезных отзывов, чтобы иметь возможность влиять на процесс разработки. Чем больше вы задерживаете выпуск важной части системы тем меньше у вас будет времени исправить ее.

Пробное решение

Создавайте пробные решения для ответа на сложные технические вопросы, для обоснования тех или иных технологических решений. При принятии любого технологического решения существует риск и пробные решения призваны уменьшить его.

Сделайте программу которая воспроизводит исследуемую проблему и игнорирует все остальное. Большинство пробных решений не предназначены для последующего использования так что ожидайте что они будут выброшены. Цель их создания - уменьшить риск принятия неправильного технического решения или более точная оценка времени на реализацию User Story.

Собрание стоя

Каждое утро проводится собрание для обсуждения проблем, их решений и для усиления концентрации команды. Собрание проводится стоя для избежания длительных дискуссий не интересных всем

членам команды.

В типичном собрании большинство участников ничего не вносят, просто участвуют чтобы послушать что скажут другие. Большое количество времени людей тратится чтобы получить небольшое количество коммуникации. Поэтому участие всех людей в собраниях уводит ресурсы из проекта и создает хаос в планировании.

Для такого рода коммуникаций и нужно собрание стоя. Намного лучше иметь одно короткое обязательное собрание, чем множество длинных на которых большинство разработчиков должно все равно присутствовать.

Если у вас проводятся ежедневные собрания стоя, то все остальные собрания должны посещать только те люди, которые необходимы и будут что-либо привносить. Более того, имеется возможность даже избежать некоторых собраний. С ограничением участников, большинство собраний может быть проведено спонтанно перед монитором, где обмен идеями намного более интенсивен.

Ежедневное утреннее собрание это не еще одна трата времени. Оно позволит вам избежать многих других собраний и сэкономит больше времени, чем на него затрачено.

Метафора Системы

Всегда выбирайте System Metaphor - простую и понятную концепцию чтобы члены команды называли все вещи одинаковыми именами. Для понимания системы и исключения дублирующего кода чрезвычайно важно как вы называете объекты. Если вы можете предположить как называется какой-либо объект в системе (если вы знаете что он делает) и он правда так называется - вы сохраните уйму времени и сил. Создайте систему имен для своих объектов так, чтобы каждый член команды мог пользоваться ею без специальных знаний о системе.

Unit Test-ы

Unit тесты играют ключевую роль в ХР. Они позволяют быстро менять код не боясь наделать новых ошибок. Unit тест пишется для каждого класса, тест должен проверять все аспекты работы класса - тестировать все что может не работать.

Хитростью здесь является то, что тест для класса должен быть написан раньше самого класса. Это значит, что как только вы выпустите первый работающий результат, он будет поддерживаться системой тестирования. Для проведения тестов пишется специальная система тестирования со своим интерфейсом.

Unit тест для класса хранится в общем репозитории вместе с кодом класса. Никакой код не может быть выпущен без Unit теста. Перед отдачей кода разработчик должен удостовериться что все тесты проходят без ошибок. Никто не может отдать код, если все не прошли 100%. Другими словами поскольку до ваших изменений все тесты проходили, то если вы имеете ошибки, то это результат ваших изменений. Вам и исправлять. Иногда бывает неправильным или неполным код теста. В таком случае надо исправлять его.

Огромным искушением является сэкономить на Unit тестах когда мало времени. Но этим Вы только обманываете себя. Чем сложнее написать тест, тем больше времени он потом сэкономит. Это доказано практикой.

Unit тесты позволяют осуществить коллективное владение кодом. Они позволяют относительно легко пересматривать плохой код. Также Unit тесты позволяют в любой момент иметь стабильно работающую систему.

User Story

User Story (что-то типа рассказа пользователя) - это описание того как система должна работать. Каждая User Story написана на карточке и представляет какой-то кусок функциональности системы,

имеющий логический смысл с точки зрения Заказчика. Форма - один-два абзаца текста понятного пользователю (не сильно технического).

User Story пишется Заказчиком. Они похожи на сценарии использования системы, но не ограничиваются пользовательским интерфейсом. По каждой истории пишутся функциональные тесты, подтверждающие что данная история корректно реализована - их еще называют приемочными (Acceptance tests). Каждой User Story дается приоритет со стороны бизнеса (пользователь, заказчик, отдел маркетинга) и оценка времени выполнения со стороны разработчиков. Каждая история разбивается на задачи и ей назначается время когда ее начнут реализовывать.

User Stories используются в XP вместо традиционных требований. Главное отличие User Story от требований (requirements) - уровень детализации. User Story содержит минимум информации, необходимой для обоснованной оценки того, сколько времени надо на ее реализацию.

Типичная User Story занимает 1-3 недели идеального времени. История требующая менее 1 недели слишком детализирована. История требующая более 3 недель может быть разбита на части - отдельные истории.

Скорость проекта

Скорость Проекта (или просто скорость) это мера того, как быстро выполняется работа в вашем проекте.

Чтобы измерить скорость проекта, вы просто должны посчитать объем User Stories, или как много (по времени) задач было выполнено за итерацию. Просто посчитайте суммарное время оценки объема работы (идеальное время).

Во время планирования релиза скорость проекта используется для оценки того сколько User Stories может быть сделано.

Во время планирования итерации, скорость проекта в предыдущей итерации используется для

определения того сколько User Stories надо планировать в текущую итерацию.

Этот простой механизм позволяет разработчикам восстановиться после трудной итерации. Если у вас после восстановления остается свободное время - идете к заказчику и просите еще User Story. В результате скорость опять возрастет.

Не надо использовать этот параметр как абсолютный. Он не подходит для сравнения продуктивности двух проектов, поскольку каждая команда имеет свои особенности. Этот параметр важен только для команды чтобы держать ее на "ровном киле".

Вы должны ожидать небольшие изменения скорости проекта во время работы. Но если скорость изменяется сильно, необходимо провести перепланирование релиза. Как только новая система уходит к пользователям, ожидайте изменения скорости проекта, так как у вас появятся задачи по поддержке.

Когда обнаружена ошибка

Если обнаруживается ошибка, то создается тест, чтобы предотвратить ее повторное появление. Ошибка, произошедшая в рабочей системе (уже установленной), требует написания функционального теста. Создание функционального теста непосредственно перед диагностикой ошибки позволяет заказчикам четко описать проблему и довести эту проблему до разработчиков.

Не выполнившийся функциональный тест требует создания Unit Test. Это помогает сфокусировать усилия по отладке и четко показывает когда ошибка исправлена.

Вам это не понадобится

Воздержитесь от заполнения системы вещами, которые понадобятся вам в будущем. Только 10% от ожидаемого действительно понадобится вам в первоначальном виде, то есть 90% вашего времени будет потрачено впустую.

Всегда есть искушение добавить функциональность сейчас а не потом, потому что мы видим как сейчас добавить ее и чувствуем что система будет намного лучше. Но мы всегда должны напоминать себе что это нам никогда не понадобится. Дополнительная функциональность только замедляет прогресс и съедает ресурсы. Забудьте про будущие требования и дополнительную гибкость. Сконцентрируйтесь на том, что необходимо сделать сейчас.

Контрольные вопросы

1. Какие из перечисленных действий являются стадиями создания ИС?

Проведение научно-исследовательских работ

Разработка технического задания

Обследование объекта

Формирование требований к ИС

2. Какие из указанных этапов создания ИС входят в стадию технического проектирования?

Разработка предварительных проектных решений по системе и её частям

Разработка и адаптация программ

Разработка и оформление документации на поставку комплектующих изделий

Разработка проектных решений по системе и её частям

3. На какой стадии создания ИС осуществляется разработка и адаптация программ?

Эскизного проектирования
Технического проектирования
Разработки рабочей документации

4. Какие из перечисленных показателей отражаются в схеме маршрута движения документов?

Действующие алгоритмы расчета показателей и возможные методы контроля
Количество документов
Действующие средства связи
Место формирования показателей документа

5. В каком разделе технического задания указываются требуемые значения производственно-экономических показателей объекта, которые должны быть достигнуты при внедрении ИС?

Требования к системе
Назначение и цели создания (развития) системы
Характеристика объектов автоматизации

6. В каком разделе технического проекта приводится обоснование выделения подсистем ИС?

Постановка задач и алгоритмы решения
Функциональная и организационная структура системы
Пояснительная записка

7. Сформулируйте цель методологии проектирования ИС

Формирование требований, направленных на обеспечение возможности комплексного использования корпоративных данных в управлении и планировании деятельности предприятия

Автоматизация ведения бухгалтерского аналитического учета и технологических процессов
Регламентация процесса проектирования ИС и обеспечение управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой ИС, так и к характеристикам процесса разработки

8. Решение каких задач обеспечивается внедрением методологии проектирования ИС?

Обеспечить нисходящее проектирование ИС (проектирование «сверху-вниз», в предположении, что одна программа должна удовлетворять потребности многих пользователей)

Гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта

Обеспечить удобную дисциплину сопровождения, модификации и наращивания системы

9. Укажите составляющие этапа проектирования ИС. + Проектирование объектов данных

Спецификация требований к приложениям

Выбор архитектуры ИС

Разработка программного кода приложений

Инсталляция базы данных

10. К какому классу ТПР относится используемая в ИС СУБД?

Элементные ТПР

Подсистемные ТПР

Объектные ТПР

11. Что отражают бизнес-правила при модельно-ориентированном проектировании?

Выполнение работ для модели бизнес-функций

Условия корректности совместного применения различных компонентов ИС и используются для поддержания целостности создаваемой системы. (формулировка не соответствует вопросу)

12. Что отражает модель функций при модельно-ориентированном проектировании?

Иерархическую декомпозицию функциональной деятельности предприятия

Иерархическую структуру подчинения подразделений и персонала

Набрано баллов

4 Анализ и моделирование функциональной области внедрения ИС

4.1 Полная бизнес-модель компании

Практика выработала ряд подходов к проведению организационного анализа, но наибольшее распространение получил инжиниринговый подход. Организационный анализ компании при таком подходе проводится по определенной схеме с помощью полной бизнес-модели компании. Компания рассматривается как целевая, открытая, социально-экономическая система, принадлежащая иерархической совокупности открытых внешних надсистем (рынок, государственные учреждения и пр.) и внутренних подсистем (отделы, цеха, бригады и пр.). Возможности компании определяются характеристиками ее структурных подразделений и организацией их взаимодействия. На рис. 4.1 представлена обобщенная схема организационного бизнес-моделирования. Построение бизнес-модели компании начинается с описания модели взаимодействия с внешней средой по закону единства и борьбы противоположностей, то есть с определения миссии компании.

Миссия согласно [ISO-15704] -это

1. деятельность, осуществляемая предприятием для того, чтобы выполнить функцию, для которой оно было учреждено, - предоставления заказчикам продукта или услуги.

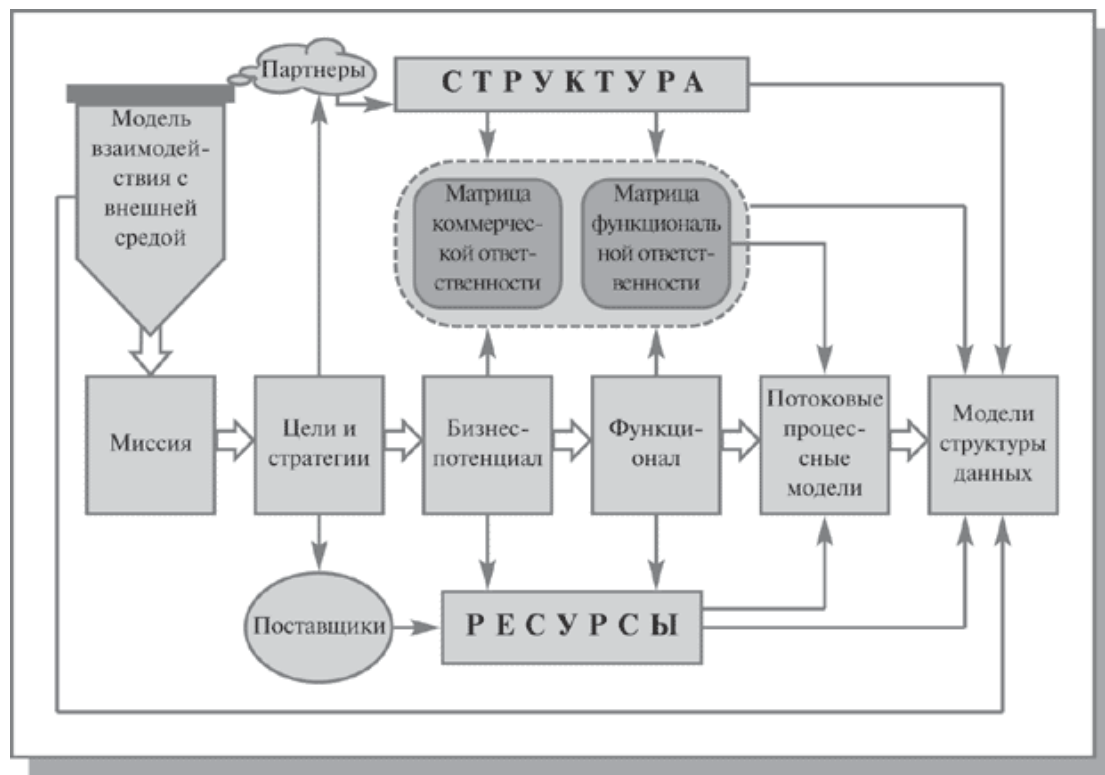


Рис. 4.1: Обобщенная схема организационного бизнес- моделирования

2. Механизм, с помощью которого предприятие реализует свои цели и задачи.

Миссия компании по удовлетворению социально-значимых потребностей рынка **определяется как компромисс интересов рынка и компании**. При этом миссия как атрибут открытой системы разрабатывается, с одной стороны, исходя из рыночной конъюнктуры и позиционирования компании относительно других участников внешней среды, а с другой - исходя из объективных возможностей компании и ее субъективных ценностей, ожиданий и принципов. Миссия **является своеобразной мерой** устремлений компании и, в частности, определяет рыночные претензии компании (предмет конкурентной борьбы). Определение миссии позволяет сформировать дерево целей компании - иерархические списки уточнения и детализации миссии.

Дерево целей формирует дерево стратегий - иерархические списки уточнения и детализации достижения целей. При этом на корпоративном уровне разрабатываются стратегии роста, интеграции и инвестиции бизнесов. Блок бизнес-стратегий определяет продуктовые и конкурентные стратегии, а также стратегии сегментации и продвижения. Ресурсные стратегии определяют стратегии привлечения материальных, финансовых, человеческих и информационных ресурсов. Функциональные стратегии определяют стратегии в организации компонентов управления и этапов жизненного цикла продукции. Одновременно выясняется потребность и предмет партнерских отношений (субподряд, сервисные услуги, продвижение и пр.). Это позволяет обеспечить заказчикам необходимый продукт требуемого качества, в нужном количестве, в нужном месте, в нужное время и по приемлемой цене. При этом компания может занять в партнерской цепочке создаваемых ценностей оптимальное место, где ее возможности и потенциал будут использоваться наилучшим образом. Это дает возможность сформировать бизнес-потенциал компании - набор видов коммерческой деятельности, направленный на удовлетворение потребностей конкретных сегментов рынка. Далее, исходя из специфики каналов сбыта, формируется первоначальное представление об организационной структуре (определяются центры коммерческой ответственности). Возникает понимание основных ресурсов, необходимых для воспроизводства товарной номенклатуры.

Бизнес-потенциал, в свою очередь, определяет функционал компании - перечень бизнес-функций, функций менеджмента и функций обеспечения, требуемых для поддержания на регулярной основе указанных видов коммерческой деятельности. Кроме того, уточняются необходимые для этого ресурсы (материальные, человеческие, информационные) и структура компании.

Построение бизнес-потенциала и функционала компании позволяет с помощью матрицы проекций определить **зоны ответственности менеджмента**.

Матрица проекций - модель, представленная в виде матрицы, задающей систему отношений между классификаторами в любой их комбинации.

Матрица коммерческой ответственности закрепляет ответственность структурных подразделений за получение дохода в компании от реализации коммерческой деятельности. Ее дальнейшая детализация (путем выделения центров финансовой ответственности) обеспечивает построение финансовой модели компании, что, в свою очередь, позволяет внедрить систему бюджетного управления. Матрица функциональной ответственности закрепляет ответственность структурных звеньев (и отдельных специалистов) за выполнение бизнес-функций при реализации процессов коммерческой деятельности (закупка, производство, сбыт и пр.), а также функций менеджмента, связанных с управлением этими процессами (планирование, учет, контроль в области маркетинга, финансов, управления персоналом и пр.). Дальнейшая детализация матрицы (до уровня ответственности отдельных сотрудников) позволит получить функциональные обязанности персонала, что в совокупности с описанием прав, обязанностей, полномочий обеспечит разработку пакета должностных инструкций.

Описание бизнес-потенциала, функционала и соответствующих матриц ответственности представляет собой **статическое описание компании**. При этом процессы, протекающие в компании пока в свернутом виде (как функции), идентифицируются, классифицируются и, что особенно важно, закрепляются за исполнителями (будущими хозяевами этих процессов).

На этом этапе бизнес-моделирования формируется общепризнанный набор основополагающих внутрифирменных регламентов:

- базовое Положение об организационно-функциональной структуре компании;
- пакет Положений об отдельных видах деятельности (финансовой, маркетинговой и т.д.);
- пакет Положений о структурных подразделениях (цехах, отделах, секторах, группах и т.п.);
- должностные инструкции.

Это вносит прозрачность в деятельность компании за счет четкого разграничения и документального закрепления зон ответственности менеджеров.

Дальнейшее развитие (детализация) бизнес-модели происходит на этапе динамического описания компании на уровне процессных потоковых моделей. Процессные потоковые модели - это модели, описывающие процесс последовательного во времени преобразования материальных и информационных потоков компании в ходе реализации какой-либо бизнес-функции или функции менеджмента. Сначала (на верхнем уровне) описывается логика взаимодействия участников процесса, а затем (на нижнем уровне) - технология работы отдельных специалистов на своих рабочих местах.

Завершается организационное бизнес-моделирование разработкой **модели структур данных, которая определяет перечень и форматы документов, сопровождающих процессы в компании, а также задает форматы описания объектов внешней среды, компонентов и регламентов самой компании.** При этом создается система справочников, на основании которых получают пакеты необходимых документов и отчетов.

Такой подход позволяет описать деятельность компании с помощью универсального множества управленческих регистров (цели, стратегии, продукты, функции, организационные звенья и др.).

Управленческие регистры по своей структуре представляют собой иерархические классификаторы. Объединяя классификаторы в функциональные группы и закрепляя между собой элементы различных классификаторов с помощью матричных проекций, можно получить полную бизнес-модель компании.

При этом происходит процессно-целевое описание компании, позволяющее получить взаимосвязанные ответы на следующие вопросы: зачем-что-где-кто-как-когда-кому-сколько (рис. 4.2).

Следовательно полная бизнес-модель компании - это совокупность функционально ориентированных информационных моделей, обеспечивающая взаимосвязанные ответы на следующие вопросы: "зачем "что "где "кто "сколько "как "когда "кому"(рис. 4.3).

Таким образом, организационный анализ предполагает построение комплекса взаимосвязанных информационных моделей компании, который включает:

- **Стратегическую модель целеполагания** (отвечает на вопросы: зачем компания занимается именно этим бизнесом, почему предполагает быть конкурентоспособной, какие цели и стратегии для этого необходимо реализовать);
- **Организационно-функциональную модель** (отвечает на вопрос кто-что делает в компании и кто за что отвечает);
- **Функционально-технологическую модель** (отвечает на вопрос что-как реализуется в компании);
- **Процессно-ролевую модель** (отвечает на вопрос кто-что-как-кому);
- **Количественную модель** (отвечает на вопрос сколько необходимо ресурсов);
- **Модель структуры данных** (отвечает на вопрос в каком виде описываются регламенты компании и объекты внешнего окружения).

Представленная совокупность моделей обеспечивает необходимую полноту и точность описания компании и позволяет вырабатывать понятные требования к проектируемой информационной системе.

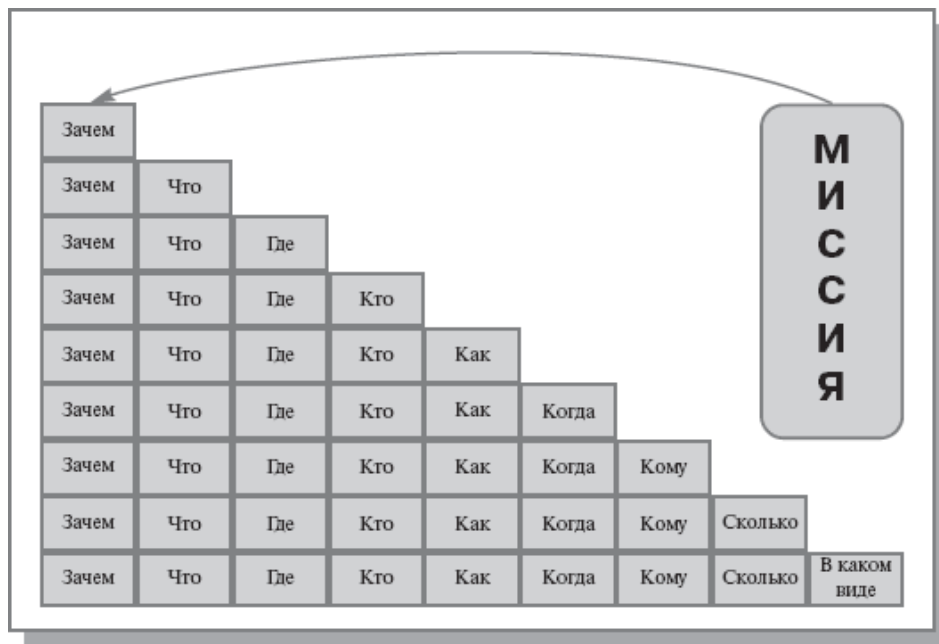


Рис. 4.2: Основные этапы процессно-целевого описания компании

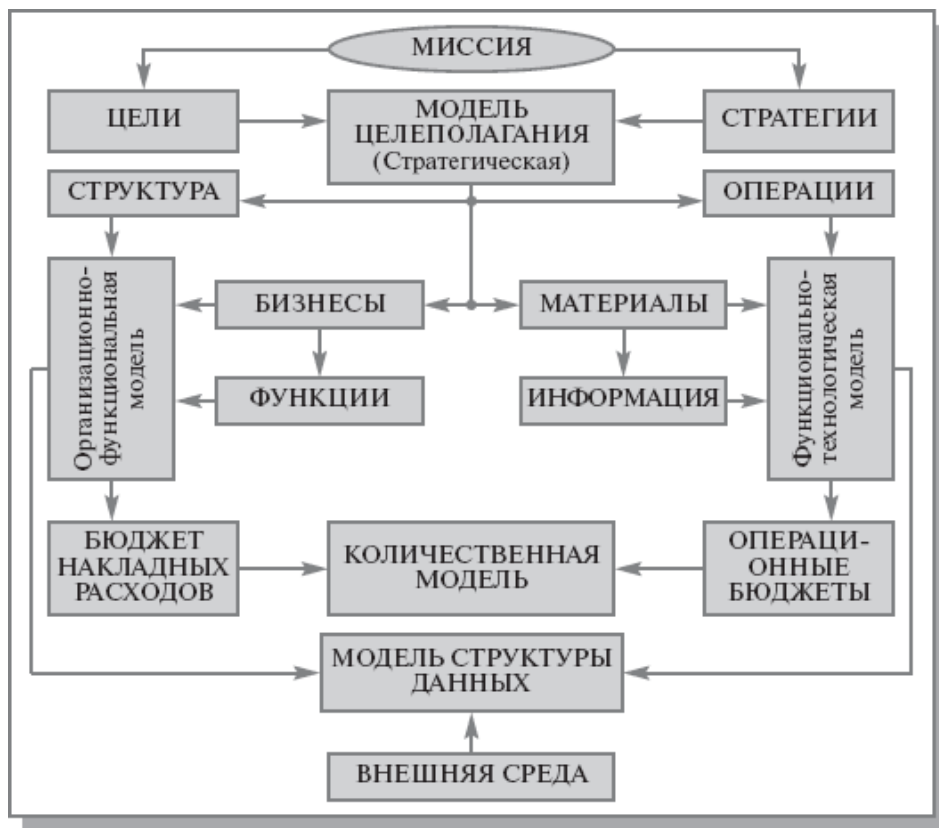


Рис. 4.3: Полная бизнес-модель компании

4.2 Шаблоны организационного бизнес-моделирования

Технология организационного бизнес-моделирования предполагает использование типовых шаблонных техник описания компании.

Шаблон разработки миссии

Как было сказано выше, любая компания с ее микро- и макроокружением представляет собой иерархию вложенных друг в друга открытых, субъектно-ориентированных систем. Компания, с одной стороны, является частью рынка, а с другой отстаивает в конкурентной борьбе собственные интересы. Миссия представляет собой результат позиционирования компании среди других участников рынка. Поэтому миссию компании нельзя описывать путем анализа ее внутреннего устройства. Для построения модели взаимодействия компании с внешней средой (определение миссии компании на рынке) необходимо:

- идентифицировать рынок (надсистему), частью которого является компания;
- определить свойства (потребности) рынка;
- определить предназначение (миссию) компании, исходя из ее роли на рынке.

Кроме этого, миссия, как было сказано выше, это компромисс между потребностями рынка, с одной стороны, и возможностями и желанием компании удовлетворить эти интересы, с другой. Поиск компромисса может быть выполнен по шаблону, представленному на рис. 4.4.

При разработке модели миссии компании рекомендуется:

1. Описать базис конкурентоспособности компании - совокупность характеристик компании как социально-экономической системы. Например:

| | | | надо | | | | |
|--------|--|---------------------------|------------------------------|---------------|-----------|------------------|------------|
| | | | рыночная конъюнк- тура | внешняя среда | | | |
| | | | | Политика | Экономика | Социал. сфера | Технология |
| объект | | Уникальность технологий | | | | | |
| | | Исключительность ресурсов | | | | | |
| | | Знания и умения | | | | | |
| хочу | | | | | | | |
| | | Ценности и ожидания | | | | | |

МИССИЯ

МИССИЯ

Рис. 4.4: Шаблон разработки миссии (матрица проекций)

- для объекта - уникальность освоенных технологий и исключительность имеющихся в компании ресурсов (финансовых, материальных, информационных и др.)
- для субъекта - знания и умения персонала и опыт менеджеров.

Это определяет уникальность ресурсов и навыков компании и формирует позицию "могу".

2. Выяснить конъюнктуру рынка, т.е. определить наличие платежеспособного спроса на предлагаемые товары или услуги и степень удовлетворения рынка конкурентами. Это позволяет понять потребности рынка и сформировать позицию "надо".
3. Выявить наличие способствующих и противодействующих факторов для выбранного вида деятельности со стороны государственных институтов в области политики и экономики.
4. Оценить перспективу развития технологии в выбранной сфере деятельности.
5. Оценить возможную поддержку или противодействие общественных организаций.
6. Сопоставить результаты вышеперечисленных действий с учетом правовых, моральных, этических и др. ограничений со стороны персонала и сформировать позицию "хочу".
7. Оценить уровень возможных затрат и доходов.
8. Оценить возможность достижения приемлемого для всех сторон компромисса и сформулировать Миссию компании в соответствии с шаблоном, приведенным на рис. 4.5.

Миссия в широком понимании представляет собой основную деловую концепцию компании, изложенную в виде восьми положений, определяющих взаимоотношения компании с другими субъектами:

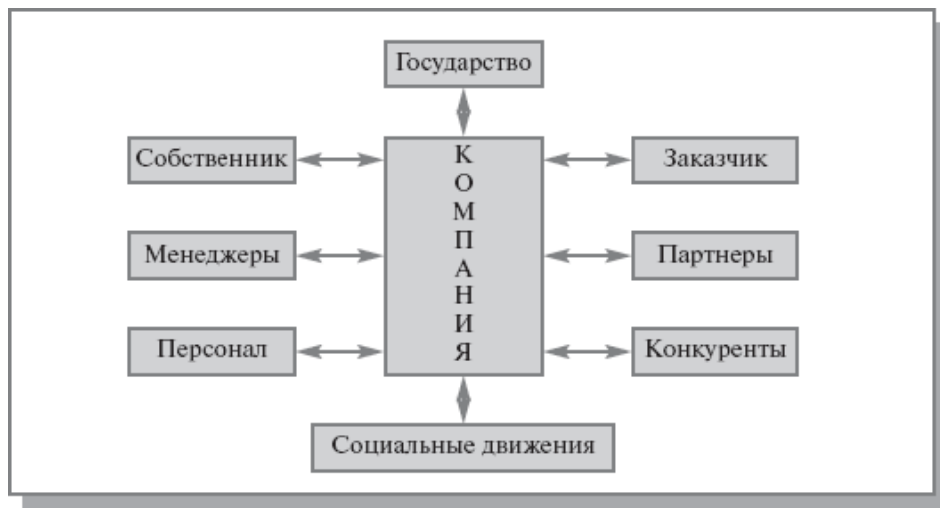


Рис. 4.5: Шаблон разработки миссии

- что получит Заказчик в части удовлетворения своих потребностей;
- кто, для чего и как может выступать в качестве партнера компании;
- на какой основе предполагается строить отношения с конкурентами (какова, в частности, готовность пойти на временные компромиссы);
- что получит собственник и акционеры от бизнеса;
- что получают от бизнеса компании менеджеры;
- что получит от компании персонал;
- в чем может заключаться сотрудничество с общественными организациями;
- как будут строиться отношения компании с государством (в частности, возможное участие в поддержке государственных программ).

Шаблон формирования бизнесов

В соответствии с разработанной Миссией компании определяются социально значимые потребности, на удовлетворение которых направлен бизнес компании.

Разработка бизнес-потенциала компании может быть выполнена по Шаблону формирования бизнесов, представленному на рис. 4.6.

В результате формируются базовый рынок и базовый продукт, детализация которых определяет предложения компании глазами покупателей (товарные группы) и однородные по отношению к продуктам компании группы покупателей (сегменты рынка). С помощью матричной проекции (рис. 4.7) устанавливается соответствие между сформированными товарными группами и сегментами рынка



Рис. 4.6: Шаблон формирования бизнесов

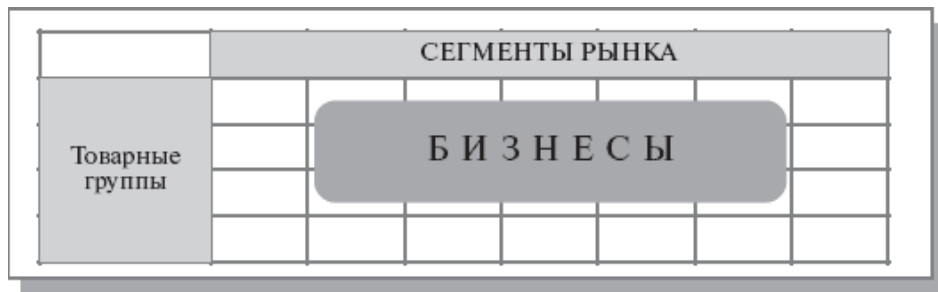


Рис. 4.7: Шаблон формирования бизнесов (матрица проекций)

и определяется список бизнесов компании (на пересечении строк и столбцов находятся бизнесы компании).

Шаблон формирования функционала компании (основных бизнес-функций)

На основании списка бизнесов, с помощью матричной проекции (рис. 4.8) формируется классификатор бизнес-функций компании.

Для формирования основных функций менеджмента компании сначала разрабатываются и утверждаются два базовых классификатора - "Компоненты менеджмента" (перечень используемых на предприятии инструментов/контуров управления) и "Этапы управленческого цикла" (технологическая цепочка операций, последовательно реализуемых менеджерами при организации работ в любом контуре управления). Далее аналогично, с помощью матрицы проекций, формируется список основных функций менеджмента. На рис. 4.9 приведены примеры классификаторов, на основании которых построена матрица - генератор основных функций менеджмента.



Рис. 4.8: Шаблон формирования основных бизнес-функций

| Компоненты менеджмента Этапы управленческого цикла | Структуры | Логистика | Финансы | Экономика | Учет | Маркетинг | Персонал |
|---|-----------|-----------|---------|-----------|------|-----------|----------|
| Сбор информации | | | | | | | |
| Выработка решений | | | | | | | |
| Реализация | | | | | | | |
| Учет | | | | | | | |
| Контроль | | | | | | | |
| Анализ | | | | | | | |
| Регулирование | | | | | | | |

**ФУНКЦИИ
МЕНЕДЖМЕНТА
(основные)**

Рис. 4.9: Шаблон формирования основных функций менеджмента

Представленные матричные проекции (рис. 4.8, рис. 4.9)) позволяют формировать функции любой степени детализации путем более подробного описания как строк, так и столбцов матрицы.

Шаблон формирования зон ответственности за функционал компании

Формирование зон ответственности за функционал компании выполняется с помощью матрицы организационных проекций (рис. 4.10).

Матрица организационных проекций представляет собой **таблицу, в строках которой расположен список исполнительных звеньев, в столбцах - список функций, выполняемых в компании.** Для каждой функции определяется исполнительное звено, отвечающее за эту функцию.

Заполнение такой таблицы позволяет по каждой функции найти исполняющие ее подразделения или сотрудника. Анализ заполненной таблицы позволяет увидеть "пробелы" как в исполнении функций, так и в загруженности сотрудников, а также рационально перераспределить все задачи между исполнителями и закрепить как систему в документе "Положение об организационной структуре".

Положение об организационной структуре - **это внутрифирменный документ, фиксирующий: продукты и услуги компании, функции, выполняемые в компании, исполнительные звенья, реализующие функции, распределение функций по звеньям.**

Таблица проекций функций на исполнительные звенья может иметь весьма большую размерность. В средних компаниях это, например, 500 единиц - 20 звеньев на 25 функций. В больших компаниях это может быть 5 000 единиц - 50 звеньев на 100 функций.

Аналогично строится матрица коммерческой ответственности.

Шаблон потокового процессного описания

Шаблон потокового процессного описания приведен на рис. 4.11. Такое описание дает представление о процессе последовательного преобразования ресурсов в продукты усилиями различных исполните-

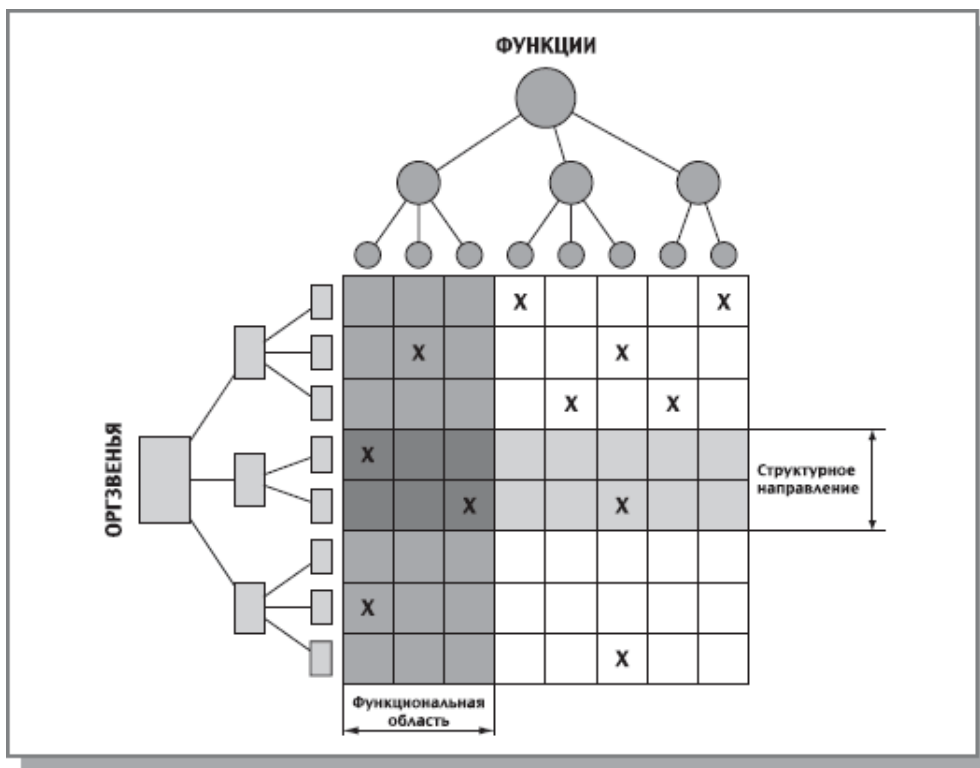


Рис. 4.10: Шаблон распределения функций по организационным звеньям

лей на основании соответствующих регламентов. Методики построения процессных моделей будут приведены ниже.

4.3 Построения организационно-функциональной модели компании

Организационно-функциональная модель компании строится на основе функциональной схемы деятельности компании рис. 4.11.

На основании миссии формируются цели и стратегии компании. С их помощью определяется необходимый набор продуктов и, как следствие - требуемые ресурсы. Воспроизводство продукции происходит за счет переработки ресурсов в основном производственном цикле. Его компоненты формируют необходимые бизнес-функции для поставки ресурсов, производства продуктов и их распределения в места реализации. Для управления указанным процессом воспроизводства формируется совокупность компонентов менеджмента, которая порождает набор функций управления. Для поддержания процессов воспроизводства и управления формируются наборы соответствующих функций обеспечения (охраны, технического оснащения, профилактики и ремонта и пр.). Такой подход позволяет описать предприятие с помощью универсального множества управленческих регистров (цели, стратегии, продукты, функции, организационные звенья и пр.). Управленческие регистры представляют собой иерархические классификаторы. Объединяя классификаторы в функциональные группы и закрепляя между собой элементы различных классификаторов с помощью матричных проекций, можно получить модель организационной структуры компании.

Для построения организационно-функциональной модели используется всего два типа элементарных моделей.

Древовидные модели (классификаторы) - точные иерархические списки выделенных объектов

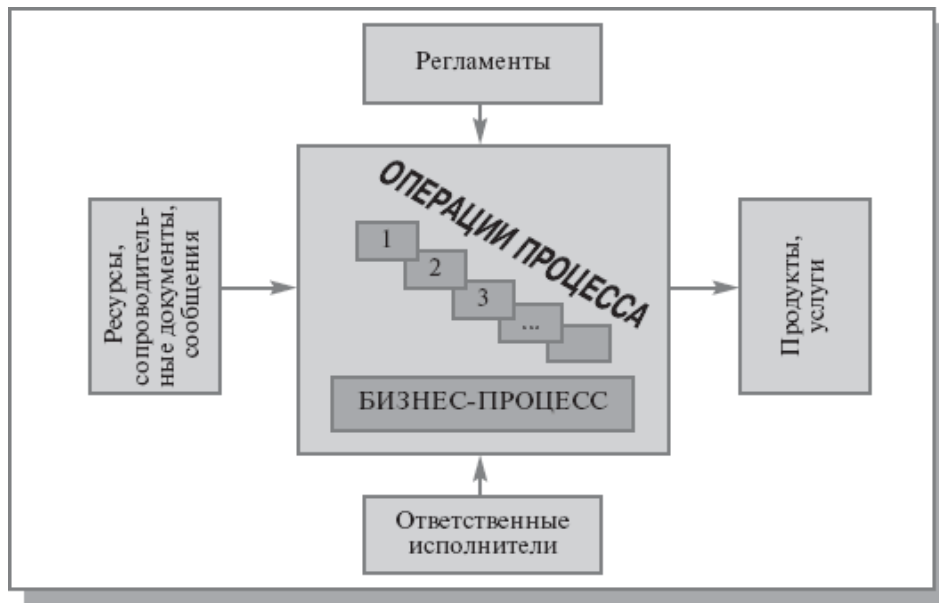


Рис. 4.11: Потоковая процессная модель

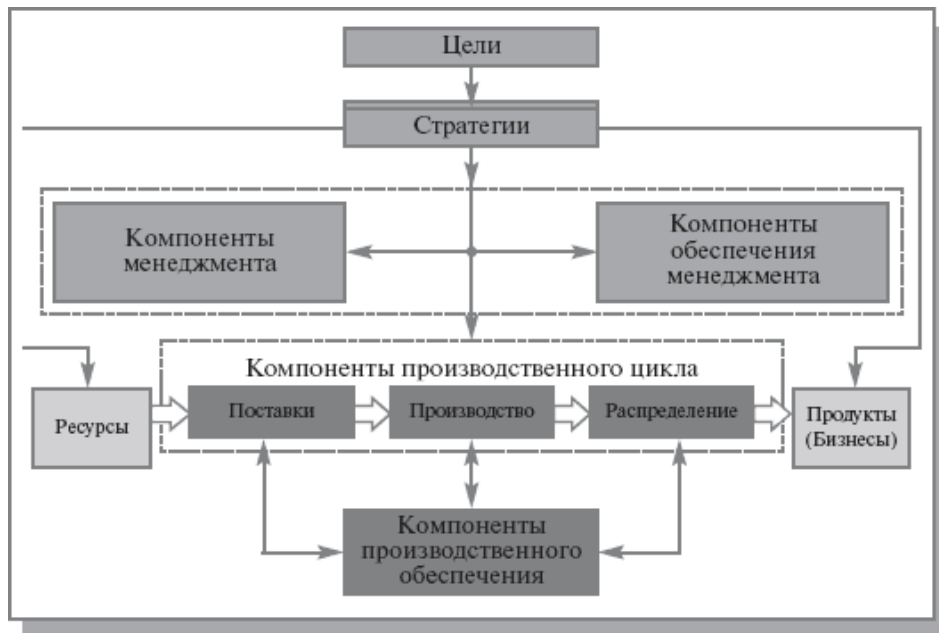


Рис. 4.12: Функциональная схема компании

управления (организационных звеньев, функций, ресурсов, в том числе исполнительных механизмов для бизнес-процессов, документов и их структуры, и т.п.). Каждый элемент классификатора может быть дополнительно охарактеризован рядом атрибутов: тип, шкала, комментарий и т.п. Фактически, классификаторы представляют собой набор управленческих регистров, содержащих, в основном, неколичественную информацию, совокупность которых задает систему координат для описания деятельности компании. Количество таких списков-классификаторов определяется целью построения модели.

Матричные модели - это проекции, задающие систему отношений между классификаторами в любой их комбинации. Связи могут иметь дополнительные атрибуты (направление, название, индекс, шкала и вес).

В начальной модели применяется всего несколько классификаторов предметной области:

- основные группы продуктов и услуг компании;
- ресурсы, потребляемые компанией в ходе своей деятельности;
- функции (процессы), поддерживаемые в компании;
- организационные звенья компании.

В классификаторе функций обычно выделяют три базовых раздела:

- основные функции - непосредственно связанные с процессом преобразования внешних ресурсов в продукцию и услуги предприятия;
- функции менеджмента - или функции управления предприятием;
- функции обеспечения - поддерживающие производственную, коммерческую и управленческую деятельность.

Главной функцией компании является предоставление продуктов и услуг, поэтому сначала производится формальное описание, согласование и утверждение руководством предприятия перечня его бизнесов (направлений коммерческой деятельности), продукции и услуг. Из этого классификатора внешним контрагентам должно быть понятно, чем предприятие интересно рынку, а для внутренних целей - для чего нужен тот или иной функционал компании.

В результате этих операций производится идентификация функционала и создается единая терминология описания функций предприятия, которая должна быть согласована всеми ведущими менеджерами. При составлении классификатора оргзвеньев важно, чтобы уровень детализации функций соответствовал уровню детализации звеньев. После формирования всех базовых классификаторов с помощью матричных проекций производится их закрепление за оргзвеньями предприятия:

Процесс формирования матрицы проекций функций на оргзвенья на практике напоминает игру в крестики-нолики (рис. 4.10).

По строчкам таблицы указываются подразделения, по столбцам - функции, составляющие содержание процесса управления или бизнес-процесса в данной компании. На пересечениях функций и подразделений, которые ответственны за выполнение функции, ставится крестик. Для проекций большой размерности используется механизм расстановки связей между двумя классификаторами, представленных списками.

Стандартная практика построения моделей организационно-функциональной структуры компаний поддерживает два уровня детализации:

1. агрегированную модель;
2. детализированную модель.

Агрегированная модель - модель организационной структуры, учетные регистры которой имеют ограничение по степени детализации до 2-3 уровней.

Целью построения данной модели является предоставление информации об организационной структуре высшим руководителям компании для проведения стратегического анализа, анализа соответствия данной структуры стратегии и внешнему окружению компании. Модель может также предоставляться внешним пользователям (например, потенциальным инвесторам как иллюстрация к бизнес-плану, крупным клиентам и др.).

Детализированная модель - модель организационной структуры, детализация учетных регистров которой производится на более глубоких уровнях, чем в агрегированной модели. Степень детализации в модели обусловлена конкретными потребностями компании (создание определенных организационных регламентов).

Целью построения данной модели является предоставление информации о распределении функциональных обязанностей между подразделениями компании, а также об организации бизнес-процессов в компании. Построение детализированной модели позволяет создавать различные внутрифирменные регламенты: Положения об организационной структуре рис. 4.13.

Ниже приведен пример описания фрагментов организационно-функциональной модели производственного предприятия рис. 4.14 и торгового предприятия рис. 4.15. Приведенные матрицы проекций являются основой для выделения бизнес-процессов предприятия и их владельцев на последующих этапах создания ИС.

Схема создания Положения об организационно- функциональной структуре компании

Функции подразделений производственного предприятия рассматриваются в рамках следующих функциональных областей:

- корпоративное управление;
- финансы;
- персонал;



Рис. 4.13: Схема создания Положения об организационно-функциональной структуре компании

| Функциональная область | Функциональная область | | | | | | | | | | |
|--|------------------------|----|----|----|----|----|----|----|----|----|----|
| | EM | FM | HR | MM | OF | OP | PD | PF | PR | QM | SL |
| Зам. ген. дир. по кадрам – инвентарь ОТК | | | | | | | | | | | |
| ОТК | | | | X | | X | | | X | X | |
| ТИЦ | | | | | | X | | | | X | |
| Химическая лаборатория | | | | | | | X | | | X | |
| Зам. ген. дир. по правовым вопросам | | | | | | | | | | | |
| Юридический отдел | X | | | | X | | | | X | | X |
| ОВЭС | | | | | | | | | | X | |
| Главный инженер | | | | | | | | | | | |
| Первый зам. гл. инж. | | | | | | | | | | | |
| ОТК | | | | | | X | | | | | |
| ОТ Мер | | | | | | X | | | X | | |
| Зам. гл. инж. | | | | | | | | | | | |
| ТОЦ | | | | | | | X | | | | |
| ЭМО | | | | | | | | | | | |
| Зам. гл. инж. по подготовке производства | | | | | | | | | | | |
| ОИТ | | | | | | X | | X | | X | |
| ОИХ | | | | | X | | | | | | |
| Зам. гл. инж. по проектированию | | | | | | | | | | | |
| Зам. ген. дир. по наладке | | | | | | | | | | | |
| Служба управления персоналом | X | | | X | | | | | | | |
| Зам. ген. дир. по ИТ | | | | | | | | | | | |
| СИТ | X | | | | | | | | | | |
| Главный бухгалтер | | | | | | | | | | | |
| Бухгалтерия | | X | X | X | X | | | | X | | |

| Функциональная область | Функциональная область | | | | | | | | | | |
|--|------------------------|----|----|----|----|----|----|----|----|----|----|
| | EM | FM | HR | MM | OF | OP | PD | PF | PR | QM | SL |
| Первый зам. ген. директора | | | | | | | | | | | |
| Начальник ОМТО | | | | | | | | | | | |
| ОМТО | | | | X | | | | X | X | X | |
| Бюро операций | | | | | | | | | | | |
| Главный инженер | | | | | | | | | | | |
| ПЛО | | X | | | X | X | | X | | | |
| Отдел отгрузки и упаковки | | | X | | | | | | | | |
| Транспортный цех | | | | X | | | | | | | |
| Производство | | | | | X | | | | | | |
| Зам. ген. дир. по маркетингу | | | | | | | | | | | |
| Служба маркетинга | | | | | | X | | | | | |
| Зам. ген. дир. по финансам и правовым вопросам | | | | | | | | | | | |
| ПЗО | | X | | X | X | | X | X | | | |
| Финансовый отдел | | | X | | | | | | | X | |
| Зам. ген. дир. по перспективному развитию | | | | | | | | | | | |
| СПИП | X | | | | | X | | | | | |
| Помощник ген. директора | | | | | | | | | | | |
| ТИЦ | | X | | | | | X | | | | |
| САК | | | | X | | X | X | X | X | X | |
| ОИПН | | | | | | X | | | | | X |

Функция выполняется отделом

Функция не выполняется отделом

Информация не указана

X Функция выполняется отделом
 Функция не выполняется отделом
 Информация не указана

Рис. 4.14: Распределение функций по подразделениям производственного предприятия

| Функциональная область | | | | | | | |
|--|----|----|----|----|----|----|----|
| | CS | EM | FM | MK | OF | PR | SL |
| Клиентский сервис | | | | | | | |
| Корпоративное управление | | | | | | | |
| Финансы | | | | | | | |
| Маркетинг | | | | | | | |
| Заказы | | | | | | | |
| Снабжение/закупки | | | | | | | |
| Сбыт/Продажи | | | | | | | |
| Генеральный директор | X | X | X | X | X | | X |
| Зам. Ген. директора по сбыту (продажи) | X | X | X | X | X | | X |
| Зам. Ген. директора по коммерческим вопросам (закупки) | | X | X | X | | X | |
| Экономист | | | X | | X | | |
| Помощник по правовым вопросам | X | X | | | X | | X |
| Начальник отдела сбыта | X | X | X | X | X | | X |
| Группа менеджеров | X | | | X | X | | |
| Отдел оформления заказов | | | X | | X | | |
| НПЦ | | | | | | | |
| Секретариат | | | | | | X | X |
| Бухгалтерия | | | X | | X | X | |

Функции выполняемые отделом, отмечены "X".

Рис. 4.15: Распределение функций по подразделениям торгового предприятия

- материальные ресурсы;
- заказы;
- производство;
- разработка продуктов;
- планирование;
- снабжение/закупки;
- качество;
- сбыт/продажи.

Распределение функций по структурным подразделениям в разрезе отдельных функциональных областей деятельности по управлению производственным предприятием представлено на рис. 4.15.

Функции подразделений торгового предприятия рассматриваются в рамках иных функциональных областей (см. рис. 4.15).

Контрольные вопросы

1. Дайте определение понятию «Миссия компании»

Деятельность, осуществляемая предприятием для того, чтобы выполнить функцию, для которой оно было учреждено, - предоставления заказчикам продукта или услуги

Дерево целей и стратегий

Механизм, с помощью которого предприятие реализует свои цели и задачи

2. Дайте определение понятию «Функционал компании»

Перечень бизнес – функций и функций менеджмента

Перечень бизнес – функций, функций менеджмента и функций обеспечения

Перечень бизнес – функций

3. Дайте определение понятию «бизнес-потенциал компании»

Набор видов коммерческой деятельности, направленный на удовлетворение потребностей конкретных сегментов рынка

Перечень бизнес-функций, функций менеджмента и функций обеспечения

Перечень бизнес – функций

4. Какая модель отвечает на вопрос *кто-что* делает в компании и *кто за что* отвечает?

Организационно-функциональная модель

Процессно-ролевая модель

Функционально-технологическая модель

5. Какая модель отвечает на вопросы: *зачем* компания занимается именно этим бизнесом, *почему* предполагает быть конкурентоспособной, *какие* цели и стратегии для этого необходимо реализовать?

Модель структуры данных

Организационно-функциональная модель

Процессно-ролевая модель

Стратегическая модель целеполагания

Функционально-технологическая модель

6. Какая модель отвечает на вопросы кто-что-как-кому?

Организационно-функциональная модель

Модель структуры данных

Стратегическая модель целеполагания

Процессно-ролевая модель

Функционально-технологическая модель

7. Какая модель определяет перечень и форматы документов, сопровождающих процессы в компании, а также задает форматы описания объектов внешней среды, компонентов и регламентов самой компании?

Функционально-технологическая модель

Модель структур данных

Количественная модель

8. Какие модели описывают процесс последовательного во времени преобразования материальных и информационных потоков компании в ходе реализации какой-либо бизнес - функции или функции менеджмента?

Модели структур данных

Процессные потоковые модели

Функциональные модели

9. Какие типы элементарных моделей используются для построения организационно-функциональной структуры?

Матричные модели

Древовидные модели (классификаторы)

Процессные модели

Набрано баллов

5 Спецификация функциональных требований к ИС

5.1 Процессные потоковые модели

Разработка требований к проектируемой ИС строится на основе статического и динамического описания компании. Статическое описание компании, рассмотренное в лекции 4, проводится на уровне функциональных моделей и включает описание бизнес-потенциала, функционала и соответствующих матриц ответственности.

Дальнейшее развитие (детализация) бизнес-модели происходит на этапе динамического описания компании на уровне процессных потоковых моделей.

Процессные потоковые модели — это модели, описывающие процесс последовательного во времени преобразования материальных и информационных потоков компании в ходе реализации какой-либо бизнес-функции или функции менеджмента. На верхнем уровне описывается логика взаимодействия участников процесса, на нижнем — технология работы отдельных специалистов на своих рабочих местах. Процессные потоковые модели отвечают на вопросы **кто—что—как—кому** (см. лекцию 4 рис. 4.13).

Современное состояние экономики характеризуется переходом от традиционной функциональной модели деятельности компании, построенной на принципах разделения труда, узкой специализации

и жестких иерархических структурах, к модели процессной, основанной на интеграции работ вокруг бизнес-процессов.

Главными недостатками функционального подхода являются:

- разбиение технологий выполнения работы на отдельные фрагменты, иногда между собой несвязанные, которые выполняются различными структурными подразделениями;
- отсутствие целостного описания технологий выполнения работы;
- сложность увязывания простейших задач в технологию, производящую реальный товар или услугу;
- отсутствие ответственности за конечный результат;
- высокие затраты на согласование, налаживание взаимодействия, контроль и т. д.;
- отсутствие ориентации на клиента.

Процессный подход предполагает смещение акцентов от управления отдельными структурными элементами на управление сквозными бизнес-процессами, связывающими деятельность всех структурных элементов. Каждый деловой процесс проходит через ряд подразделений, т. е. в его выполнении участвуют специалисты различных отделов компании. Чаще всего приходится сталкиваться с ситуацией, когда собственно процессами никто не управляет, а управляют лишь подразделениями. Более того, структура компаний строится без учета возможностей оптимизации деловых процессов, обеспечивающих необходимые функции. Процессный подход позволяет устранить фрагментарность в работе, организационные и информационные разрывы, дублирование, нерациональное использование финансовых, материальных и кадровых ресурсов.

Процессный подход к организации деятельности предприятия предполагает:

- широкое делегирование полномочий и ответственности исполнителям;
- сокращение уровней принятия решений;
- сочетание принципа целевого управления с групповой организацией труда;
- повышенное внимание к вопросам обеспечения качества;
- автоматизация технологий выполнения бизнес-процессов.

Согласно стандарту «Основные Положения и Словарь — ИСО/ОПМС 9000:2000» (п. 2.4) понятие «Процессный подход» определяется как:

«Любая деятельность, или комплекс деятельности, в которой используются ресурсы для преобразования входов в выходы, может рассматриваться как процесс. Чтобы результативно функционировать, организации должны определять и управлять многочисленными взаимосвязанными и взаимодействующими процессами. Часто выход одного процесса образует непосредственно вход следующего. Систематическая идентификация и менеджмент применяемых организацией процессов, и особенно взаимодействия таких процессов, могут считаться «процессным подходом».

Основной принцип процессного подхода определяет структурирование бизнес-системы в соответствии с деятельностью и бизнес-процессами предприятия, а не в соответствии с его организационно-штатной структурой. Именно бизнес-процессы, обеспечивающие значимый для потребителя результат, представляют ценность и для специалистов, проектирующих ИС. Процессная модель компании должна строиться с учетом следующих положений:

1. Верхний уровень модели должен отражать только контекст диаграммы – взаимодействие моделируемого единственным контекстным процессом предприятия с внешним миром.

2. На втором уровне должны быть отражены тематически сгруппированные бизнес-процессы предприятия и их взаимосвязи.
3. Каждая из деятельности должна быть детализирована на бизнес-процессы.
4. Детализация бизнес-процессов осуществляется посредством бизнес – функций.
5. Описание элементарной бизнес–операции осуществляется с помощью миниспецификации.

Процессный подход требует комплексного изучения различных сторон жизни организации — правовых основ и правил деятельности, организационной структуры, функций и показателей результатов их исполнения, интерфейсов, ресурсного обеспечения, организационной культуры. В результате анализа создается модель деятельности «как есть». Обработка этой модели с помощью различных аналитических методов позволяет проверить, насколько деловые процессы рациональны, а также определить, является ли та или иная операция ориентированной на общественно значимый конечный результат или излишней бюрократической процедурой.

В ходе анализа деловых процессов детально исследуются сферы ответственности подразделений ведомства, его руководителей и сотрудников. Это позволяет установить адреса владельцев деловых процессов, в результате чего процессы перестают быть бесхозными, создаются условия для разработки и внедрения систем стимулирования и ответственности за конечные результаты, определяются моменты и процедуры передачи ответственности. Анализ и оценка деловых процессов позволяют подойти к обоснованию стандартов их выполнения, допустимых рисков и диапазонов свободы принятия решений исполнителями, предельных нормативов затрат ресурсов на единицу эффекта.

Однако чисто «процессная компания» является скорее иллюстрацией правильной организации работ. В действительности все бизнес-процессы компании протекают в рамках организационной структуры предприятия, описывающей функциональные компетентности и отношения.

Управление всей текущей деятельностью компании ведется по двум направлениям — управление функциональными областями, которые поддерживают множество унифицированных бизнес-процессов, разделенных на операции, и управление интегрированными бизнес-процессами, задачей которого является маршрутизация и координация унифицированных процессов для выполнения как оперативных заказов потребителей, так и глобальных проектов самой организации (рис. 5.1).

Фактически основной задачей организационного проектирования является выбор оптимального соотношения между эффективностью использования ресурсов и эффективностью процессов. Жесткая специализация подразделений экономит ресурсы организации, но снижает качество реализации процессов. Создание «процессных» команд, включающих собственных специалистов по всем ключевым операциям, обходится достаточно дорого, но при этом значительно сокращается время и повышается точность выполнения процесса. Иногда организации могут позволить себе выбрать этот путь, особенно в тех случаях, когда создается высокая ценность процесса, за которую потребитель согласен платить. Но, как правило, ищется какой-то компромисс на основе процессно-матричных структур. Когда компания начинает ориентироваться на процессы, исключительно важной становится роль владельцев интегрированных межфункциональных процессов, касающихся многих функциональных областей. Кроме того, новая парадигма деятельности предприятия вызывает появление большого числа процессов управления, распределенных по всему предприятию, а не сосредоточенных в специализированных организационных единицах: это системы качества, бюджетирования, маркетинга и т.п. Поэтому постановка бюджетирования как организационной, а не только финансовой задачи предполагает делегирование полномочий, т.е. власти (с которой нелегко расстаться). На более низкие уровни делегируется ответственность за принятие финансовых решений: о заключении сделки-договора, об оплате, о закупке, о скидках и отпуске в кредит и т.п. Это позволяет упростить связи между подразделениями и снизить количество уровней вертикального прохождения документов, т.е. является необходимым условием реализации классической схемы реинжиниринга. Таким образом, процессная ориентация ведет к перестройке организационной структуры, делает организационную

| | Функционал ная область 1 | Функционал ная область 2 | Функционал ная область 3 | Функционал ная область N |
|--|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
| Процесс 1 (⇒ операции, ↑↑ исполнитель) | ↑↑ ⇒ | → | ↑↑ ⇒ | ↑↑ ⇒ |
| Процесс 2 | | | | ↑↑ ⇒ |
| Процесс 2 | | ↑↑ ⇒ | ↗ | |
| Процесс 1 (⇒ операции, ↑↑ исполнитель) | ↑↑ ⇒ | | | |

Рис. 5.1: Схема управления деятельностью компании

структуру компании более «плоской», что иллюстрирует тесную связь между «вертикальным» описанием организации (как структуры распределения ответственности, полномочий и взаимоотношений) и ее «горизонтальным» описанием, как системы процессов.

5.2 Основные элементы процессного подхода

В рамках процессного подхода любое предприятие рассматривается как бизнес-система – система, которая представляет собой связанное множество бизнес-процессов, конечными целями которых является выпуск продукции или услуг.

Под бизнес-процессом понимают совокупность различных видов деятельности, которые создают результат, имеющий ценность для потребителя. Бизнес-процесс – это цепочка работ (функций), результатом которой является какой-либо продукт или услуга.

Каждый бизнес-процесс имеет свои границы (подробнее см. лекции 6, 7) и роли.

В процессном подходе используются следующие ключевые роли:

Владелец процесса – человек, отвечающий за ход и результаты процесса в целом. Он должен знать бизнес-процесс, следить за его выполнением и совершенствовать его эффективность. Владельцу бизнес-процесса необходимо обладать коммуникативностью, энтузиазмом, способностью влиять на людей и производить изменения.

Лидер команды — работник, обладающий знаниями о бизнес-процессе и имеющий позитивные личные качества.

Коммуникатор – работник, обучающий команду различным методам работы, подготавливающий совместно с лидером совещания и анализирующий их результат.

Координатор процесса – работник, отвечающий за согласованную работу всех частей бизнеса и обеспечивающий связь с другими бизнес-процессами. Координатор должен обладать административ-

ными способностями и пониманием стратегических целей предприятия.

Участники команды – специалисты различных уровней иерархии. Участники команды получают поддержку и методическое обеспечение от консультанта и коммуникатора, вместе с лидером проводят моделирование, анализ и оценку бизнес-процесса.

Одним из основных элементов процессного подхода является команда. Существует несколько типов процессных команд:

Ситуационная команда – обычно работает на постоянной основе и выполняет периодически повторяющуюся работу.

Виртуальная команда – создается для разработки нового продукта или услуги.

Ситуационный менеджер – высококвалифицированный специалист, способный самостоятельно выполнить до 90% объема работ.

Важной задачей процессного подхода является формирование процессных команд. Подготовка и формирование команды включает:

- учебные курсы;
- практический тренинг по освоению методов, методик и др.;
- психологическое тестирование;
- тестирование рабочих навыков.

Достижение определенной совокупности целей за счет выполнения бизнес-процессов называется деревом целей. Дерево целей имеет, как правило, иерархический вид. Каждая цель имеет свой вес и критерий (количественный или качественный) достижимости.

Бизнес-процессы реализуют бизнес-функции предприятия. Под бизнес-функцией понимают вид деятельности предприятия. Множество бизнес-функций представляет иерархическую декомпозицию функциональной деятельности и называется деревом функций.

Бизнес-функции связаны с показателями деятельности предприятия, образующими дерево показателей. На основании показателей строится система показателей оценки эффективности выполнения процессов. Владельцы процессов контролируют свои бизнес-процессы с помощью данной системы показателей. Наиболее общими показателями оценки эффективности бизнес-процессов являются:

- количество производимой продукции заданного качества за определенный интервал времени;
- количество потребляемой продукции;
- длительность выполнения типовых операций и др.

5.3 Выделение и классификация процессов

При процессном описании должны решаться, как минимум, две задачи:

1. Идентификация всей системы «функциональных областей» и процессов компании и их взаимосвязей.
2. Выделение «ключевых» интегрированных процессов и их описание на потоковом уровне.

Каждая деятельность компании реализуется как процесс, который имеет своего потребителя: внешнего — клиента или внутреннего — сотрудников или подразделения компании, реализующих

другие процессы. На стадии системного описания процессов и выявляется значимость каждого процесса — в том числе происходит очищение от малопонятной деятельности. На этом этапе выбираются **ключевые процессы** для потокового описания, которое необходимо, например, для создания информационной системы предприятия.

Наиболее распространены следующие четыре вида бизнес-процессов:

1. Процессы, создающие наибольшую добавленную стоимость (экономическую стоимость, которая определяется издержками компании, относимыми на продукцию).
2. Процессы, создающие наибольшую ценность для клиентов (маркетинговую стоимость за счет дифференциации продукции).
3. Процессы с наиболее интенсивным межзвенным взаимодействием, создающие транзакционные издержки.
4. Процессы, определенные стандартами ИСО 9000, как обязательные к описанию при постановке системы менеджмента качества.

Важнейшим шагом при структуризации любой компании является выделение и классификация бизнес-процессов. Целесообразно основываться на следующих классах процессов:

1. основные;
2. процессы управления;
3. процессы обеспечения;
4. сопутствующие;



Рис. 5.2: Упрощенная модель деятельности компании

- 5. вспомогательные;
- 6. процессы развития.

Рассмотрим модель деятельности компании (рис. 5.2), при описании которой используют процессы управления, основные бизнес-процессы и процессы обеспечения.

Основные бизнес-процессы — это процессы, ориентированные на производство товаров и услуг, представляющие ценность для клиента и обеспечивающие получение дохода.

Основные процессы образуют «жизненный цикл» продукции компании. Критериями эффективности таких процессов являются обычно качество, точность и своевременность выполнения каждого заказа. Многие потребители рассматривают увеличение качества как нечто более важное, чем уменьшение цены. Искусный продавец может получить заказ на выполнение работ в условиях конкуренции с другими фирмами, однако только качество товара или услуги определяет в большей степени, повторит ли потребитель свой заказ у этого продавца еще раз. Таких процессов, при развитой деятельности компании, может быть много. Все они описываются по производственно-коммерческим

цепочкам: «первичное взаимодействие с клиентом и определение его потребностей реализация запроса (заявки, заказа, контракта и т.п.) послепродажное сопровождение и мониторинг удовлетворения потребностей». Процесс «реализации (запроса клиента)» может быть декомпозирован на следующие подпроцессы — процессы более низкого уровня:

- разработка (проектирование) продукции;
- закупка (товаров, материалов, комплектующих изделий);
- транспортировка (закупленного);
- разгрузка, приемка на склад и хранение (закупленного);
- производство (со своим технологическим циклом и внутренней логистикой);
- приемка на склад и хранение (готовой продукции);
- отгрузка (консервация и упаковка, погрузка, доставка);
- пуско-наладка;
- оказание услуг (предусмотренных контрактом на поставку или имеющих самостоятельное значение) и т.п.

Эти этапы цепочки также достаточно стандартны (например, в стандарте ИСО редакции 1994 г. приведены многие из этих процессов в качестве обязательных и подлежащих сертификации). Проверить, какие бизнес-цепочки существуют на предприятии, можно с помощью проекции каждого из выделенных «бизнесов, продукции и услуг» на вышеуказанный (стандартный) библиотечный классификатор жизненного или уже производственного цикла.

Для оценки этапов работы с любым документом можно использовать также анализ «жизненного цикла документа», который может выглядеть следующим образом:

- предоставляет исходные данные;
- подготавливает, разрабатывает;
- заполняет;
- корректирует;
- оформляет;
- подписывает;
- контролирует соответствие установленным требованиям;
- визирует;
- согласует;
- утверждает;
- акцентирует (принимает к сведению, использует);
- хранит;
- снимает копию.

Здесь тоже может быть применена своя матрица-генератор, как средство проверки полноты, — идентификация цикла.

Можно также воспользоваться референтными моделями деятельности аналогичных компаний — они могут сопоставляться с процессами конкурентов, лидеров отрасли, а также совершенствоваться.

Процессы управления – это процессы, охватывающие весь комплекс функций управления на уровне каждого бизнес-процесса и бизнес-системы в целом. Процессы управления имеют своей целью выработку и принятие управленческого решения. Данные управленческие решения могут приниматься относительно всей организации в целом, отдельной функциональной области или отдельных процессов, например:

- стратегическое управление;
- организационное проектирование (структуризация);
- маркетинг;
- финансово-экономическое управление;
- логистика и организация процессов;
- менеджмент качества;
- персонал.

Другая возможная **систематизация функций управления** связана с понятием управленческого цикла и базируется на пяти исходных функциях управления: планирование, организация, распоряительство, координация, контроль. Самая распространенная ошибка — это смешение этих принципов.

Для реализации процессного описания исключительно важным является то, что любая управленческая деятельность разворачивается по так называемому «управленческому циклу», который включает:

- сбор информации;
- выработку решения;
- реализацию;
- учет;
- контроль;
- анализ;
- регулирование.

Например, наиболее часто встречающиеся варианты детализации:

- сбор информации;
- определение состава собираемой информации;
- определение форм отчетности.
- выработка решения;
- анализ альтернатив;

- подготовка вариантов решения;
- принятие решения;
- выработка критериев оценки;
- реализация;
- планирование;
- организация;
- мотивация;
- координация;
- контроль исполнения
- учет результатов;
- сравнение по принятым критериям;
- анализ
- анализ дополнительной информации;
- диагностика возможных причин отклонений;
- регулирование

- регулирование на уровне реализации (возврат к п.3);
- регулирование на уровне выработки решения (возврат к п.1,2)

Каждый из этих этапов имеет своих характерных для него исполнителей — управленцев, которых можно отнести к трем основным категориям:

- руководитель (ответственный за принятие и организацию выполнения решений);
- специалист-аналитик (ответственный за подготовку решения и анализ отклонений);
- технические исполнители (сбор информации, учет, коммуникации).

Согласно некоторым подходам, в процессах управления выделяются два типа процессов, относящихся, соответственно, к двум типам менеджмента, условно обозначаемым как «менеджмент ресурсов» и «менеджмент организации», которые отличаются по объекту управления, базовым моделям и, что важно для описания процессов, — своими управленческими циклами. Тогда модель деятельности предприятия становится двухуровневой (рис. 5.3)

Из этой модели следует, что сами циклы ресурсного планирования нуждаются в регламентации — то есть ресурсное управление может осуществляться только по специально разработанным организационным регламентам.

В основе **цикла управления ресурсами** лежит расчет или имитационное моделирование и контроль результатов:

- выбор (или получение от системы верхнего уровня) целевого критерия оценки качества решения;
- сбор информации о ресурсах предприятия или возможностях внешней среды;



Рис. 5.3: Двухуровневая модель деятельности предприятия

- просчет вариантов (с различными предположениями о возможных значениях параметров);
- выбор оптимального варианта — принятие решения (= ресурсного плана);
- учет результатов (и отчетность);
- сравнение с принятым критерием оценки (= контроль результатов);
- анализ причин отклонений и регулирование (возврат к 1, 2 или 3).

В основе цикла организационного менеджмента лежит структурное или процессное моделирование и процедурный контроль:

- определение состава задач (обособленных функций, операций);

- выбор исполнителей (- распределение зон и степени ответственности);
- проектирование процедур (последовательности и порядка исполнения);
- согласование и утверждение регламента исполнения (- процесса, плана мероприятий);
- отчетность об исполнении;
- контроль исполнения (- процедурный контроль);
- анализ причин отклонений и регулирование (возврат к 1, 2 или 3).

Таким образом, на определенных шагах декомпозиции предприятию надо определить, какие стадии управленческого цикла реализуются по каждой из ранее выделенных задач управления. Это можно проверить с помощью матрицы-генератора, которая раскладывает компоненты менеджмента по этапам управленческого цикла.

Процессы обеспечения – это процессы, предназначенные для жизнеобеспечения основных и сопутствующих процессов и ориентированные на поддержку их универсальных средств. Например, процесс финансового обеспечения, процесс обеспечения кадрами, процесс юридического обеспечения — это вторичные процессы. Они создают и поддерживают необходимые условия для выполнения основных функций и функций менеджмента. Клиенты обеспечивающих процессов находятся внутри компании.

На верхнем уровне детализации можно выделить примерно следующие стандартные процессы обеспечения:

- обеспечение производства;
- техобслуживание и ремонт оборудования;

- обеспечение теплоэнергоресурсами;
- обслуживание и ремонт зданий и сооружений;
- технологическое обеспечение;
- метрологическое;
- техника безопасности;
- экологический контроль и т.п.
- обеспечение управления;
- информационное обеспечение;
- обеспечение документооборота;
- коммуникационное обеспечение;
- юридическое обеспечение;
- обеспечение безопасности;
- материально-техническое обеспечение управления;
- хозяйственное обеспечение;
- обеспечение коммунальными услугами;

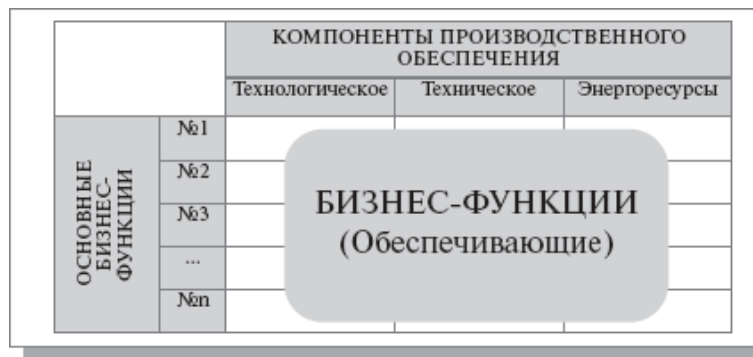


Рис. 5.4: Упрощенная матрица-генератор обеспечивающих бизнес-функций

- транспортное обслуживание и т.п.

Для каждого из выделенных выше подпроцессов также следует определить, какой основной или управленческий процесс является потребителем этих «внутренних» услуг. Для этого существуют свои матрицы-генераторы. Их можно построить отдельно для основных процессов (рис. 5.4) и процессов управления (рис. 5.4).

Разбиение данных процессов производится по индивидуальным технологическим цепочкам. Многие из обеспечивающих процессов стандартны для всех компаний или определенных видов деятельности: промышленность, торговля, предоставление услуг и т.п. Однако, как правило, данный класс функций в меньшей степени «подвергается» потоковому процессному описанию. Большинство из них достаточно хорошо регламентируются должностными и специальными инструкциями.

| Компоненты обеспечения менеджмента | | Информационные ресурсы | Внутренние коммуникации | Компьютерная обработка | Обработка средствами оргтехники | Защита информации | Телекоммуникация и сеть | Транспорт |
|------------------------------------|-----|------------------------|-------------------------|------------------------|---------------------------------|-------------------|-------------------------|-----------|
| Основные функции менеджмента | №1 | | | | | | | |
| | № 2 | | | | | | | |
| | № 3 | | | | | | | |
| | ... | | | | | | | |
| | №n | | | | | | | |

**ФУНКЦИИ
МЕНЕДЖМЕНТА
(обеспечивающие)**

Рис. 5.5: Матрица-генератор обеспечивающих бизнес-функций

5.4 Референтная модель бизнес-процесса

В качестве основного каркаса, объединяющего и систематизирующего все знания по бизнес-модели, можно использовать референтную модель. Референтная модель — это модель эффективного бизнес-процесса, созданная для предприятия конкретной отрасли, внедренная на практике и предназначенная для использования при разработке/реорганизации бизнес-процессов на других предприятиях. По сути, референтные модели представляют собой эталонные схемы организации бизнеса, разработанные для конкретных бизнес-процессов на основе реального опыта внедрения в различных компаниях по всему миру. Они включают в себя проверенные на практике процедуры и методы организации управления. Референтные модели позволяют предприятиям начать разработку собственных моделей на базе уже готового набора функций и процессов.

Референтная модель бизнес-процесса представляет собой совокупность логически взаимосвязанных функций. Для каждой функции указывается исполнитель, входные и выходные документы или информационные объекты. Элементы (функции и документы) референтной модели бизнес-процесса содержат ссылки на соответствующие объекты ИС, а также документы и другую информацию (пользовательские инструкции, ответственных разработчиков), расположенную в репозитории проекта. Отсюда и название — референтная модель (в переводе с английского ссылочная модель).

5.5 Проведение предпроектного обследования предприятий

Обследование предприятия является важным и определяющим этапом проектирования ИС. Длительность обследования обычно составляет 1-2 недели. В течение этого времени системный аналитик должен обследовать не более 2-3 видов деятельности (учет кадров, бухгалтерия, перевозки, маркетинг и др.).

Сбор информации для построения полной бизнес-модели организации часто сводится к изучению документированных информационных потоков и функций подразделений, а также производится путем интервьюирования и анкетирования.

К началу работ по обследованию организация обычно предоставляет комплект документов, в состав которого обычно входят:

1. Сводная информация о деятельности предприятия.
 - а) Информация об управленческой, финансово-экономической, производственной деятельности предприятия.
 - б) Сведения об учетной политике и отчетности.
2. Регулярный документооборот предприятия.
 - а) Реестр входящей информации.
 - б) Реестр внутренней информации.
 - с) Реестр исходящей информации.
3. Сведения об информационно-вычислительной инфраструктуре предприятия.
4. Сведения об ответственных лицах.

Списки вопросов для интервьюирования и анкетирования составляются по каждому обследуемому подразделению и утверждаются руководителем компании. Это делается с целью:

- предотвращения доступа к конфиденциальной информации;

- усиления целевой направленности обследования;
- минимизации отвлечения сотрудников предприятий от выполнения должностных обязанностей.

Общий перечень вопросов (с их последующей детализацией) включает следующие пункты:

- основные задачи подразделений;
- собираемая и регистрируемая информация;
- отчетность;
- взаимодействие с другими подразделениями.

Анкеты для руководителей и специалистов могут содержать следующие вопросы:

- Каковы (с позиций вашего подразделения) должны быть цели создания интегрированной системы управления предприятием?
- Организационная структура подразделения.
- Задачи подразделения.
- Последовательность действий при выполнении задач.
- С какими типами внешних организаций (банк, заказчик, поставщик и т.п.) взаимодействует подразделение и какой информацией обменивается?
- Каким справочным материалом вы пользуетесь?

- Сколько времени (в минутах) вы тратите на исполнение основных операций? На какие даты приходится «пиковые нагрузки»? (периодичность в месяц, квартал, год и т.д.) Техническое оснащение подразделения (компьютеры, сеть, модем и т.п.). Используемые программные продукты для автоматизации бизнес-процессов.
- Какие отчеты и как часто вы готовите для руководства? Ключевые специалисты подразделения, способные ответить на любые вопросы по бизнес-процессам, применяемым в подразделении.
- Характеристики удаленных объектов управления.
- Документооборот на рабочем месте.

Собранные таким образом данные, как правило, не охватывают всех существенных сторон организационной деятельности и обладают высокой степенью субъективности. И самое главное, что такого рода обследования не выявляют устойчивых факторов, связанных со специфическими особенностями организации, воздействовать на которые можно исключительно методами функциональной настройки организационной системы.

Анализ опросов руководителей обследуемых организаций и предприятий показывает, что их представления о структуре организации, общих и локальных целях функционирования, задачах и функциях подразделений, а также подчиненности работников иногда имеют противоречивый характер. Кроме того, эти представления подчас расходятся с официально декларируемыми целями и правилами или противоречат фактической деятельности.

Если структуру информационных потоков можно выявить по образцам документов и конфигурациям компьютерных сетей и баз данных, то структура реальных микропроцессов, осуществляемых персоналом в информационных контактах (в значительной мере недокументированных) остается неизвестной. Ответы на эти вопросы может дать структурно-функциональная диагностика, основанная на методах сплошной (или выборочной) фотографии рабочего времени персонала. Цель ди-

агностики — получение достоверного знания об организации и организационных отношениях ее функциональных элементов. В связи с этим к важнейшим задачам функциональной диагностики организационных структур относятся:

- классификация субъектов функционирования (категорий и групп работников);
- классификация элементов процесса функционирования (действий, процедур);
- классификация направлений (решаемых проблем), целей функционирования;
- классификация элементов информационных потоков;
- проведение обследования деятельности персонала организации;
- исследование распределения (по времени и частоте) организационных характеристик: процедур, контактов персонала, направлений деятельности, элементов информационных потоков — по отдельности и в комбинациях друг с другом по категориям работников, видам процедур и их направлениям (согласно результатам и логике исследований);
- выявление реальной структуры функциональных, информационных, иерархических, временных, проблемных отношений между руководителями, сотрудниками и подразделениями;
- установление структуры распределения рабочего времени руководителей и персонала относительно функций, проблем и целей организации;
- выявление основных технологий функционирования организации (информационных процессов, включая и недокументированные), их целеполагания в сравнении с декларируемыми целями организации;

- выявление однородных по специфике деятельности, целевой ориентации и реальной подчиненности групп работников, формирование реальной модели организационной структуры и сравнение ее с декларируемой;
- определение причин рассогласования декларируемой и реальной структуры организационных отношений.

Сплошной «фотографией» рабочего времени называется непрерывное наблюдение и регистрация характеристик работников в процессе функционирования в течение всего рабочего дня. При этом индицируемые параметры последовательно вносятся в заранее заготовленную рабочую таблицу. Ниже представлена форма рабочей таблицы системного аналитика;

Сразу по окончании процедуры обследования таблица пополняется дополнительными характеристиками: технологическая ветвь, системная функция, предмет, аспект, эмоциональный фон и др.

Часть показателей, те, что помечены звездочкой, заполняются в процессе обследования, остальные — после. Содержание записей следующее:

- номер (по порядку);
- агент (должность обследуемого работника);
- время, в течение которого выполнялась процедура;
- процедура (наименование содержания совокупности элементарных действий, объединенных общностью решаемой частной задачи);
- содержание (суть процедуры, которая должна быть классифицирована);
- информация (направление движения информации между агентом и контрагентом);

- инициатива (инициатор начала выполнения данной процедуры);
- контрагент (должность работника, который находится с обследуемым в контакте);
- отношение (отражающая субординацию агента и контрагента форма взаимодействия в данной процедуре);
- проблема (словесная характеристика решаемой проблемы).

5.6 Результаты предпроектного обследования

Результатом предпроектного обследования является «Отчет об экспресс-обследовании предприятия», структура которого приведена ниже.

1. Краткое схематичное описание бизнес-процессов:
 - управление закупками и запасами;
 - управление производством;
 - управление продажами;
 - управление финансовыми ресурсами.
2. Основные требования и приоритеты автоматизации.
3. Оценка необходимых для обеспечения проекта ресурсов заказчика.
4. Оценка возможности автоматизации, предложения по созданию автоматизированной системы с оценкой примерных сроков и стоимости.

Операции бизнес-процесса

Проведение предпроектного обследования позволяет решить следующие задачи:

- предварительное выявление требований к будущей системе;
- определение структуры организации;
- определение перечня целевых функций организации;
- анализ распределения функций по подразделениям и сотрудникам;
- выявление функциональных взаимодействий между подразделениями, информационных потоков внутри подразделений и между ними, внешних информационных воздействий;
- анализ существующих средств автоматизации организации.

Информация, полученная в результате предпроектного обследования, анализируется с помощью методов структурного и/или объектного анализа, о которых будет сказано ниже, и используется для построения моделей деятельности организации. Модель организации предполагает построение двух видов моделей:

- модели «как есть», отражающей существующее на момент обследования положение дел в организации и позволяющей понять, каким образом функционирует данная организация, а также выявить узкие места и сформулировать предложения по улучшению;
- модели «как должно быть», отражающей представление о новых технологиях работы организации. Каждая из моделей включает в себя полную функциональную и информационную модель деятельности организации, а также модель, описывающую динамику поведения организации (в случае необходимости).

Контрольные вопросы

1. Дайте определение понятию «Процессы управления»

Процессы, охватывающие весь комплекс функций управления на уровне каждого бизнес-процесса

Процессы, охватывающие комплекс функций управления бизнес-системы в целом

Процессы, предназначенные для жизнеобеспечения основных и сопутствующих процессов и ориентированные на поддержку их универсальных средств.

2. Дайте определение понятию «Основные бизнес-процессы»

Процессы, охватывающие весь комплекс функций управления на уровне каждого бизнес-процесса и бизнес-системы в целом

Процессы, обеспечивающие получение дохода

Процессы, ориентированные на производство товаров и услуг

3. Дайте определение понятию «Процессы обеспечения»

Процессы, предназначенные для жизнеобеспечения основных процессов

Процессы, предназначенные для жизнеобеспечения основных и сопутствующих процессов и ориентированные на поддержку их универсальных средств

Процессы, обеспечивающие получение дохода

4. Какая модель отражает существующее на момент обследования положение дел в организации?

Модель «как должно быть»

Референтная модель

Модель «как есть»

5. Какая модель отражает представление о новых технологиях работы организации?

Модель «как есть»

Референтная модель

Модели «как должно быть»

6. Какая модель представляет собой эталонные схемы организации бизнеса, разработанные для конкретных бизнес-процессов?

Референтная модель

Модель «как есть»

Модель «как должно быть»

7. Каким методом обследования достигается регистрация характеристик работников в процессе функционирования в течение всего рабочего дня?

Сплошная "фотография" рабочего времени

Анкетирование

Интервьюирование

8. Какую информацию можно получить по образцам документов и конфигурациям баз данных?

Информацию о структуре информационных потоков

Информацию о структуре организации

Информацию о структуре реальных микропроцессов

9. Каким способом производится сбор информации для построения полной бизнес-модели организации?

Путем изучения документированных информационных потоков и функций подразделений

Путем интервьюирования

Путем анкетирования

Набрано баллов

6 Методологии моделирования предметной области

6.1 Структурная модель

В основе проектирования ИС лежит моделирование предметной области. Для того чтобы получить адекватный предметной области проект ИС в виде системы правильно работающих программ, необходимо иметь целостное, системное представление модели, которое отражает все аспекты функционирования будущей информационной системы. При этом под моделью предметной области понимается некоторая система, имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

Предварительное моделирование предметной области позволяет сократить время и сроки проведения проектировочных работ и получить более эффективный и качественный проект. Без проведения моделирования предметной области велика вероятность допущения большого количества ошибок в решении стратегических вопросов, приводящих к экономическим потерям и высоким затратам на последующее перепроектирование системы. Вследствие этого все современные технологии проектирования ИС основываются на использовании методологии моделирования предметной области.

К моделям предметных областей предъявляются следующие требования:

- формализация, обеспечивающая однозначное описание структуры предметной области;
- понятность для заказчиков и разработчиков на основе применения графических средств отображения модели;
- реализуемость, подразумевающая наличие средств физической реализации модели предметной области в ИС;
- обеспечение оценки эффективности реализации модели предметной области на основе определенных методов и вычисляемых показателей.

Для реализации перечисленных требований, как правило, строится **система моделей**, которая отражает структурный и оценочный аспекты функционирования предметной области.

Структурный аспект предполагает построение:

- объектной структуры, отражающей состав взаимодействующих в процессах материальных и информационных объектов предметной области;
- функциональной структуры, отражающей взаимосвязь функций (действий) по преобразованию объектов в процессах;
- структуры управления, отражающей события и бизнес-правила, которые воздействуют на выполнение процессов;
- организационной структуры, отражающей взаимодействие организационных единиц предприятия и персонала в процессах;
- технической структуры, описывающей топологию расположения и способы коммуникации комплекса технических средств.

Для отображения структурного аспекта моделей предметных областей в основном используются графические методы, которые должны гарантировать представление информации о компонентах системы. Главное требование к графическим методам документирования — простота. Графические методы должны обеспечивать возможность структурной декомпозиции спецификаций системы с максимальной степенью детализации и согласований описаний на смежных уровнях декомпозиции.

С моделированием непосредственно связана проблема **выбора языка** представления проектных решений, позволяющего как можно больше привлекать будущих пользователей системы к ее разработке. Язык моделирования – это нотация, в основном графическая, которая используется для описания проектов. Нотация представляет собой совокупность графических объектов, используемых в модели. Нотация является синтаксисом языка моделирования. Язык моделирования, с одной стороны, должен делать решения проектировщиков понятными пользователю, с другой стороны, предоставлять проектировщикам средства достаточно формализованного и однозначного определения проектных решений, подлежащих реализации в виде программных комплексов, образующих целостную систему программного обеспечения.

Графическое изображение нередко оказывается наиболее емкой формой представления информации. При этом проектировщики должны учитывать, что графические методы документирования не могут полностью обеспечить декомпозицию проектных решений от постановки задачи проектирования до реализации программ ЭВМ. Трудности возникают при переходе от этапа анализа системы к этапу проектирования и в особенности к программированию.

Главный **критерий адекватности структурной модели** предметной области заключается в функциональной полноте разрабатываемой ИС.

Оценочные аспекты моделирования предметной области связаны с разрабатываемыми показателями эффективности автоматизируемых процессов, к которым относятся:

- время решения задач;

- стоимостные затраты на обработку данных;
- надежность процессов;
- косвенные показатели эффективности, такие, как объемы производства, производительность труда, оборачиваемость капитала, рентабельность и т.д.

Для расчета показателей эффективности, как правило, используются статические методы функционального стоимостного анализа (АВС) и динамические методы имитационного моделирования.

В основе различных методологий моделирования предметной области ИС лежат принципы последовательной детализации абстрактных категорий. Обычно модели строятся на трех уровнях: на внешнем уровне (**определении требований**), на концептуальном уровне (**спецификации требований**) и внутреннем уровне (**реализации требований**). Так, на внешнем уровне модель отвечает на вопрос, что должна делать система, то есть определяется состав основных компонентов системы: объектов, функций, событий, организационных единиц, технических средств. **На концептуальном уровне** модель отвечает на вопрос, как должна функционировать система? Иначе говоря, определяется характер взаимодействия компонентов системы одного и разных типов. На внутреннем уровне модель отвечает на вопрос: с помощью каких программно-технических средств реализуются требования к системе? С позиции жизненного цикла ИС описанные уровни моделей соответственно строятся на этапах анализа требований, логического (технического) и физического (рабочего) проектирования. Рассмотрим особенности построения моделей предметной области на трех уровнях детализации.

Объектная структура

Объект — это сущность, которая используется при выполнении некоторой функции или операции (преобразования, обработки, формирования и т.д.). Объекты могут иметь динамическую или ста-

тическую природу: динамические объекты используются в одном цикле воспроизводства, например заказы на продукцию, счета на оплату, платежи; статические объекты используются во многих циклах воспроизводства, например, оборудование, персонал, запасы материалов.

На внешнем уровне детализации модели выделяются основные виды материальных объектов (например, сырье и материалы, полуфабрикаты, готовые изделия, услуги) и основные виды информационных объектов или документов (например, заказы, накладные, счета и т.д.).

На концептуальном уровне построения модели предметной области уточняется состав классов объектов, определяются их атрибуты и взаимосвязи. Таким образом строится обобщенное представление структуры предметной области.

Далее концептуальная модель на внутреннем уровне отображается в виде файлов базы данных, входных и выходных документов ЭИС. Причем динамические объекты представляются единицами переменной информации или документами, а статические объекты — единицами условно-постоянной информации в виде списков, номенклатур, ценников, справочников, классификаторов. Модель базы данных как постоянно поддерживаемого информационного ресурса отображает хранение условно-постоянной и накапливаемой переменной информации, используемой в повторяющихся информационных процессах.

Функциональная структура

Функция (операция) представляет собой некоторый преобразователь входных объектов в выходные. Последовательность взаимосвязанных по входам и выходам функций составляет бизнес-процесс. Функция бизнес-процесса может порождать объекты любой природы (материальные, денежные, информационные). Причем бизнес-процессы и информационные процессы, как правило, неразрывны, то есть функции материального процесса не могут осуществляться без информационной поддержки. Например, отгрузка готовой продукции осуществляется на основе документа "Заказ который, в свою

очередь, порождает документ "Накладная сопровождающий партию отгруженного товара.

Функция может быть представлена одним действием или некоторой совокупностью действий. В последнем случае каждой функции может соответствовать некоторый процесс, в котором могут существовать свои подпроцессы, и т.д., пока каждая из подфункций не будет представлять некоторую недекомпозируемую последовательность действий.

На внешнем уровне моделирования определяется список основных бизнес-функций или видов бизнес-процессов. Обычно таких функций насчитывается 15–20.

На концептуальном уровне выделенные функции декомпозируются и строятся иерархии взаимосвязанных функций.

На внутреннем уровне отображается структура информационного процесса в компьютере: определяются иерархические структуры программных модулей, реализующих автоматизируемые функции.

Структура управления

В совокупности функций бизнес-процесса возможны альтернативные или циклические последовательности в зависимости от различных условий протекания процесса. Эти условия связаны с происходящими событиями во внешней среде или в самих процессах и с образованием определенных состояний объектов (например, заказ принят, отвергнут, отправлен на корректировку). События вызывают выполнение функций, которые, в свою очередь, изменяют состояния объектов и формируют новые события, и т.д., пока не будет завершен некоторый бизнес-процесс. Тогда последовательность событий составляет конкретную реализацию бизнес-процесса.

Каждое событие описывается с двух точек зрения: информационной и процедурной. Информационно событие отражается в виде некоторого сообщения, фиксирующего факт выполнения некоторой функции изменения состояния или появления нового. Процедурно событие вызывает выполнение

новой функции, и поэтому для каждого состояния объекта должны быть заданы описания этих вызовов. Таким образом, события выступают в связующей роли для выполнения функций бизнес-процессов.

На внешнем уровне определяются список внешних событий, вызываемых взаимодействием предприятия с внешней средой (платежи налогов, процентов по кредитам, поставки по контрактам и т.д.), и список целевых установок, которым должны соответствовать бизнес-процессы (регламент выполнения процессов, поддержка уровня материальных запасов, уровень качества продукции и т.д.).

На концептуальном уровне устанавливаются бизнес-правила, определяющие условия вызова функций при возникновении событий и достижении состояний объектов.

На внутреннем уровне выполняется формализация бизнес-правил в виде триггеров или вызовов программных модулей.

Организационная структура

Организационная структура представляет собой совокупность организационных единиц, как правило, связанных иерархическими и процессными отношениями. Организационная единица — это подразделение, представляющее собой объединение людей (персонала) для выполнения совокупности общих функций или бизнес-процессов. В функционально-ориентированной организационной структуре организационная единица выполняет набор функций, относящихся к одной функции управления и входящих в различные процессы. В процессно-ориентированной структуре организационная единица выполняет набор функций, входящих в один тип процесса и относящихся к разным функциям управления.

На внешнем уровне строится структурная модель предприятия в виде иерархии подчинения организационных единиц или списков взаимодействующих подразделений.

На концептуальном уровне для каждого подразделения задается организационно-штатная структура должностей (ролей персонала).

На внутреннем уровне определяются требования к правам доступа персонала к автоматизируемым функциям информационной системы.

Техническая структура

Топология определяет территориальное размещение технических средств по структурным подразделениям предприятия, а коммуникация — технический способ реализации взаимодействия структурных подразделений.

На внешнем уровне модели определяются типы технических средств обработки данных и их размещение по структурным подразделениям.

На концептуальном уровне определяется способы коммуникаций между техническими комплексами структурных подразделений: физическое перемещение документов, машинных носителей, обмен информацией по каналам связи и т.д.

На внутреннем уровне строится модель "клиент-серверной" архитектуры вычислительной сети.

Описанные модели предметной области нацелены на проектирование отдельных компонентов ИС: данных, функциональных программных модулей, управляющих программных модулей, программных модулей интерфейсов пользователей, структуры технического комплекса. Для более качественного проектирования указанных компонентов требуется построение моделей, увязывающих различные компоненты ИС между собой. В простейшем случае в качестве таких моделей взаимодействия могут использоваться матрицы перекрестных ссылок: "объекты-функции" "функции-события" "организационные единицы — функции" "организационные единицы — объекты" "организационные единицы — технические средства" и т.д. Такие матрицы не наглядны и не отражают особенности реализации взаимодействий.

Для правильного отображения взаимодействий компонентов ИС важно осуществлять совместное моделирование таких компонентов, особенно с содержательной точки зрения объектов и функций. Методология структурного системного анализа существенно помогает в решении таких задач.

Структурным анализом принято называть метод исследования системы, которое начинается с ее общего обзора, а затем детализируется, приобретая иерархическую структуру с все большим числом уровней. Для таких методов характерно: разбиение на уровни абстракции с ограниченным числом элементов (от 3 до 7); ограниченный контекст, включающий только существенные детали каждого уровня; использование строгих формальных правил записи; последовательное приближение к результату. Структурный анализ основан на двух базовых принципах – "разделяй и властвуй" и принципе иерархической упорядоченности. Решение трудных проблем путем их разбиения на множество меньших независимых задач (так называемых "черных ящиков") и организация этих задач в древовидные иерархические структуры значительно повышают понимание сложных систем. Определим ключевые понятия структурного анализа.

Операция – элементарное (неделимое) действие, выполняемое на одном рабочем месте.

Функция – совокупность операций, сгруппированных по определенному признаку.

Бизнес-процесс — связанная совокупность функций, в ходе выполнения которой потребляются определенные ресурсы и создается продукт (предмет, услуга, научное открытие, идея), представляющая ценность для потребителя.

Подпроцесс – это бизнес-процесс, являющийся структурным элементом некоторого бизнес-процесса и представляющий ценность для потребителя.

Бизнес-модель – структурированное графическое описание сети процессов и операций, связанных с данными, документами, организационными единицами и прочими объектами, отражающими существующую или предполагаемую деятельность предприятия.

Существуют различные методологии структурного моделирования предметной области, среди которых следует выделить **функционально-ориентированные и объектно-ориентированные мето-**

дологии.

6.2 Функционально-ориентированные и объектно-ориентированные методологии

Процесс бизнес-моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что представляет собой моделируемая организация. В соответствии с различными представлениями об организации методики принято делить на объектные и функциональные (структурные).

Объектные методики рассматривают моделируемую организацию как набор взаимодействующих объектов – производственных единиц. Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Целью применения данной методики является выделение объектов, составляющих организацию, и распределение между ними ответственностей за выполняемые действия.

Функциональные методики, наиболее известной из которых является методика IDEF, рассматривают организацию как набор функций, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы. Основное отличие от объектной методики заключается в четком отделении функций (методов обработки данных) от самих данных.

С точки зрения бизнес-моделирования каждый из представленных подходов обладает своими преимуществами. Объектный подход позволяет построить более устойчивую к изменениям систему, лучше соответствует существующим структурам организации. Функциональное моделирование хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена. Подход от выполняемых функций интуитивно лучше понимается

исполнителями при получении от них информации об их текущей работе.

Функциональная методика IDEF0

Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Technique). Исторически IDEF0 как стандарт был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий, которая носила обозначение ICAM (Integrated Computer Aided Manufacturing). Семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF=Icam DEFinition), и последняя его редакция была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (NIST).

Целью методики является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы.

В основе методологии лежат четыре основных понятия: **функциональный блок, интерфейсная дуга, декомпозиция, глоссарий**.

Функциональный блок (Activity Box) представляет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, "производить услуги"). На диаграмме функциональный блок изображается прямоугольником (рис. 6.1). Каждая из четырех сторон функционального блока имеет свое определенное значение (роль), при этом:

- верхняя сторона имеет значение "Управление"(Control);
- левая сторона имеет значение "Вход"(Input);
- правая сторона имеет значение "Выход"(Output);



Рис. 6.1: Функциональный блок

- нижняя сторона имеет значение "Механизм"(Mechanism).

Интерфейсная дуга (Arrow) отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, представленную данным функциональным блоком. Интерфейсные дуги часто называют потоками или стрелками.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

В зависимости от того, к какой из сторон функционального блока подходит данная интерфейсная дуга, она носит название "входящей" "исходящей" или "управляющей".

Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь, по крайней мере, одну управляющую интерфейсную дугу и одну исходящую. Это и понятно – каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и

должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Декомпозиция (Decomposition) является основным понятием стандарта IDEF0. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Последним из понятий IDEF0 является глоссарий (Glossary). Для каждого из элементов IDEF0 — диаграмм, функциональных блоков, интерфейсных дуг — существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом. Этот набор называется глоссарием и является описанием сущности данного элемента. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Модель IDEF0 всегда начинается с представления системы как единого целого — одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется **контекстной диаграммой**.

В пояснительном тексте к контекстной диаграмме должна быть указана **цель** (Purpose) построения диаграммы в виде краткого описания и зафиксирована **точка зрения** (Viewpoint).

Определение и формализация цели разработки IDEF0-модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь.

Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.

Выделение подпроцессов. В процессе декомпозиции функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы, и называется дочерней (Child Diagram) по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется дочерним блоком – Child Box). В свою очередь, функциональный блок — предок называется родительским блоком по отношению к дочерней диаграмме (Parent Box), а диаграмма, к которой он принадлежит – родительской диаграммой (Parent Diagram). Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. В каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок или исходящие из него, фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0–модели.

Иногда отдельные интерфейсные дуги высшего уровня не имеет смысла продолжать рассматривать на диаграммах нижнего уровня, или наоборот — отдельные дуги нижнего отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. Для решения подобных задач в стандарте IDEF0 предусмотрено понятие туннелирования. Обозначение "туннеля" (Arrow Tunnel) в виде двух круглых скобок вокруг начала интерфейсной дуги обозначает, что эта дуга не была унаследована от функционального родительского блока и появилась (из "туннеля") только на этой диаграмме. В свою очередь, такое же обозначение вокруг конца (стрелки) интерфейсной дуги в непосредственной близости от блока–приемника означает тот факт, что

в дочерней по отношению к этому блоку диаграмме эта дуга отображаться и рассматриваться не будет. Чаще всего бывает, что отдельные объекты и соответствующие им интерфейсные дуги не рассматриваются на некоторых промежуточных уровнях иерархии, – в таком случае они сначала "погружаются в туннель" а затем при необходимости "возвращаются из туннеля".

Обычно IDEF0-модели несут в себе сложную и концентрированную информацию, и для того, чтобы ограничить их перегруженность и сделать удобочитаемыми, в стандарте приняты соответствующие ограничения сложности.

Рекомендуется представлять на диаграмме от трех до шести функциональных блоков, при этом количество подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг предполагается не более четырех.

Стандарт IDEF0 содержит набор процедур, позволяющих разрабатывать и согласовывать модель большой группой людей, принадлежащих к разным областям деятельности моделируемой системы. Обычно процесс разработки является итеративным и состоит из следующих условных этапов:

- Создание модели группой специалистов, относящихся к различным сферам деятельности предприятия. Эта группа в терминах IDEF0 называется авторами (Authors). Построение первоначальной модели является динамическим процессом, в течение которого авторы опрашивают компетентных лиц о структуре различных процессов, создавая модели деятельности подразделений. При этом их интересуют ответы на следующие вопросы: Что поступает в подразделение "на входе"?
 - Какие функции и в какой последовательности выполняются в рамках подразделения?
 - Кто является ответственным за выполнение каждой из функций?
 - Чем руководствуется исполнитель при выполнении каждой из функций?
 - Что является результатом работы подразделения (на выходе)?

На основе имеющихся положений, документов и результатов опросов создается черновик (Model Draft) модели.

- Распространение черновика для рассмотрения, согласований и комментариев. На этой стадии происходит обсуждение черновика модели с широким кругом компетентных лиц (в терминах IDEF0 — читателей) на предприятии. При этом каждая из диаграмм черновой модели письменно критикуется и комментируется, а затем передается автору. Автор, в свою очередь, также письменно соглашается с критикой или отвергает ее с изложением логики принятия решения и вновь возвращает откорректированный черновик для дальнейшего рассмотрения. Этот цикл продолжается до тех пор, пока авторы и читатели не придут к единому мнению.
- Официальное утверждение модели. Утверждение согласованной модели происходит руководителем рабочей группы в том случае, если у авторов модели и читателей отсутствуют разногласия по поводу ее адекватности. Окончательная модель представляет собой согласованное представление о предприятии (системе) с заданной точки зрения и для заданной цели.

Наглядность графического языка IDEF0 делает модель вполне читаемой и для лиц, которые не принимали участия в проекте ее создания, а также эффективной для проведения показов и презентаций. В дальнейшем на базе построенной модели могут быть организованы новые проекты, нацеленные на производство изменений в модели.

Функциональная методика потоков данных

Целью методики является построение модели рассматриваемой системы в виде диаграммы потоков данных (Data Flow Diagram — DFD), обеспечивающей правильное описание выходов (отклика системы в виде данных) при заданном воздействии на вход системы (подаче сигналов через внешние

интерфейсы). Диаграммы потоков данных являются основным средством моделирования функциональных требований к проектируемой системе.

При создании диаграммы потоков данных используются четыре основных понятия: **потоки данных, процессы (работы) преобразования входных потоков данных в выходные, внешние сущности, накопители данных (хранилища)**.

Потоки данных являются абстракциями, используемыми для моделирования передачи информации (или физических компонент) из одной части системы в другую. Потоки на диаграммах изображаются именованными стрелками, ориентация которых указывает направление движения информации.

Назначение **процесса** (работы) состоит в продуцировании выходных потоков из входных в соответствии с действием, задаваемым именем процесса. Имя процесса должно содержать глагол в неопределенной форме с последующим дополнением (например, "получить документы по отгрузке продукции"). Каждый процесс имеет уникальный номер для ссылок на него внутри диаграммы, который может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

Хранилище (накопитель) данных позволяет на указанных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилище представляет "срезы" потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее получения, при этом данные могут выбираться в любом порядке. Имя хранилища должно определять его содержимое и быть существительным.

Внешняя сущность представляет собой материальный объект вне контекста системы, являющейся источником или приемником системных данных. Ее имя должно содержать существительное, например, "склад товаров". Предполагается, что объекты, представленные как внешние сущности, не должны участвовать ни в какой обработке.

Кроме основных элементов, в состав DFD входят словари данных и миниспецификации.

Словари данных являются каталогами всех элементов данных, присутствующих в DFD, включая групповые и индивидуальные потоки данных, хранилища и процессы, а также все их атрибуты.

Миниспецификации обработки — описывают DFD-процессы нижнего уровня. Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами: множество всех миниспецификаций является полной спецификацией системы.

Процесс построения DFD начинается с создания так называемой основной диаграммы типа "звезда" на которой представлен моделируемый процесс и все внешние сущности, с которыми он взаимодействует. В случае сложного основного процесса он сразу представляется в виде декомпозиции на ряд взаимодействующих процессов. Критериями сложности в данном случае являются: наличие большого числа внешних сущностей, многофункциональность системы, ее распределенный характер. Внешние сущности выделяются по отношению к основному процессу. Для их определения необходимо выделить поставщиков и потребителей основного процесса, т.е. все объекты, которые взаимодействуют с основным процессом. На этом этапе описание взаимодействия заключается в выборе глагола, дающего представление о том, как внешняя сущность использует основной процесс или используется им. Например, основной процесс – "учет обращений граждан" внешняя сущность – "граждане" описание взаимодействия – "подаёт заявления и получает ответы". Этот этап является принципиально важным, поскольку именно он определяет границы моделируемой системы.

Для всех внешних сущностей строится таблица событий, описывающая их взаимодействие с основным потоком. Таблица событий включает в себя наименование внешней сущности, событие, его тип (типичный для системы или исключительный, реализующийся при определенных условиях) и реакцию системы.

На следующем шаге происходит декомпозиция основного процесса на набор взаимосвязанных процессов, обменивающихся потоками данных. Сами потоки не конкретизируются, определяется лишь характер взаимодействия. Декомпозиция завершается, когда процесс становится простым, т.е.:

1. процесс имеет два-три входных и выходных потока;

2. процесс может быть описан в виде преобразования входных данных в выходные;
3. процесс может быть описан в виде последовательного алгоритма.

Для простых процессов строится миниспецификация – формальное описание алгоритма преобразования входных данных в выходные.

Миниспецификация удовлетворяет следующим требованиям: для каждого процесса строится одна спецификация; спецификация однозначно определяет входные и выходные потоки для данного процесса; спецификация не определяет способ преобразования входных потоков в выходные; спецификация ссылается на имеющиеся элементы, не вводя новые; спецификация по возможности использует стандартные подходы и операции.

После декомпозиции основного процесса для каждого подпроцесса строится аналогичная таблица внутренних событий.

Следующим шагом после определения полной таблицы событий выделяются **потоки данных**, которыми обмениваются процессы и внешние сущности. Простейший способ их выделения заключается в анализе таблиц событий. События преобразуются в потоки данных от инициатора события к запрашиваемому процессу, а реакции – в обратный поток событий. После построения входных и выходных потоков аналогичным образом строятся внутренние потоки. Для их выделения для каждого из внутренних процессов выделяются поставщики и потребители информации. Если поставщик или потребитель информации представляет процесс сохранения или запроса информации, то вводится хранилище данных, для которого данный процесс является интерфейсом.

После построения потоков данных диаграмма должна быть проверена на полноту и непротиворечивость. Полнота диаграммы обеспечивается, если в системе нет "повисших" процессов, не используемых в процессе преобразования входных потоков в выходные. Непротиворечивость системы обеспечивается выполнением наборов формальных правил о возможных типах процессов: на диаграмме не может быть потока, связывающего две внешние сущности – это взаимодействие удаляется из

рассмотрения; ни одна сущность не может непосредственно получать или отдавать информацию в хранилище данных – хранилище данных является пассивным элементом, управляемым с помощью интерфейсного процесса; два хранилища данных не могут непосредственно обмениваться информацией – эти хранилища должны быть объединены.

К преимуществам методики DFD относятся:

- возможность однозначно определить внешние сущности, анализируя потоки информации внутри и вне системы;
- возможность проектирования сверху вниз, что облегчает построение модели "как должно быть";
- наличие спецификаций процессов нижнего уровня, что позволяет преодолеть логическую незавершенность функциональной модели и построить полную функциональную спецификацию разрабатываемой системы.

К недостаткам модели отнесем: необходимость искусственного ввода управляющих процессов, поскольку управляющие воздействия (потоки) и управляющие процессы с точки зрения DFD ничем не отличаются от обычных; отсутствие понятия времени, т.е. отсутствие анализа временных промежутков при преобразовании данных (все ограничения по времени должны быть введены в спецификациях процессов).

Объектно-ориентированная методика

Принципиальное отличие между функциональным и объектным подходом заключается в способе декомпозиции системы. Объектно-ориентированный подход использует объектную декомпозицию, при этом статическая структура описывается в терминах **объектов и связей** между ними, а поведение

системы описывается в терминах **обмена сообщениями** между объектами. Целью методики является построение бизнес-модели организации, позволяющей перейти от модели сценариев использования к модели, определяющей отдельные объекты, участвующие в реализации бизнес-функций.

Концептуальной основой объектно-ориентированного подхода является объектная модель, которая строится с учетом следующих принципов:

- абстрагирование;
- инкапсуляция;
- модульность;
- иерархия;
- типизация;
- параллелизм;
- устойчивость.

Основными понятиями объектно-ориентированного подхода являются объект и класс.

Объект — предмет или явление, имеющее четко определенное поведение и обладающие состоянием, поведением и индивидуальностью. Структура и поведение схожих объектов определяют общий для них класс. **Класс – это множество объектов, связанных общностью структуры и поведения.** Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм. Понятие **полиморфизм** может быть интерпретировано как способность класса принадлежать более чем одному типу. Наследование означает построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой информационной системы от стадии формирования требований до стадии реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

Большинство существующих методов объектно-ориентированного подхода включают язык моделирования и описание процесса моделирования. **Процесс** – это описание шагов, которые необходимо выполнить при разработке проекта. В качестве языка моделирования объектного подхода используется унифицированный язык моделирования UML, который содержит стандартный набор диаграмм для моделирования.

Диаграмма (Diagram) — это графическое представление множества элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями) и представляет собой некоторую проекцию системы.

Объектно-ориентированный подход обладает следующими преимуществами:

- Объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей.
- Объектная декомпозиция позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем.
- Объектная модель естественна, поскольку ориентированна на человеческое восприятие мира.

К недостаткам объектно-ориентированного подхода относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки

двух–трех проектов и накопления повторно используемых компонентов. Диаграммы, отражающие специфику объектного подхода, менее наглядны.

Сравнение существующих методик

В **функциональных моделях** (DFD-диаграммах потоков данных, SADT-диаграммах) главными структурными компонентами являются функции (операции, действия, работы), которые на диаграммах связываются между собой потоками объектов.

Несомненным достоинством функциональных моделей является реализация структурного подхода к проектированию ИС по принципу "сверху-вниз когда каждый функциональный блок может быть декомпозирован на множество подфункций и т.д., выполняя, таким образом, модульное проектирование ИС. Для функциональных моделей характерны процедурная строгость декомпозиции ИС и наглядность представления.

При функциональном подходе объектные модели данных в виде ER-диаграмм "объект — свойство — связь" разрабатываются отдельно. Для проверки корректности моделирования предметной области между функциональными и объектными моделями устанавливаются взаимно однозначные связи.

Главный недостаток функциональных моделей заключается в том, что процессы и данные существуют отдельно друг от друга — помимо функциональной декомпозиции существует структура данных, находящаяся на втором плане. Кроме того, не ясны условия выполнения процессов обработки информации, которые динамически могут изменяться.

Перечисленные недостатки функциональных моделей снимаются в **объектно-ориентированных моделях**, в которых главным структурообразующим компонентом выступает класс объектов с набором функций, которые могут обращаться к атрибутам этого класса.

Для классов объектов характерна иерархия обобщения, позволяющая осуществлять **наследование** не только атрибутов (свойств) объектов от вышестоящего класса объектов к нижестоящему классу,

но и функций (методов).

В случае наследования функций можно абстрагироваться от конкретной реализации процедур (**абстрактные типы данных**), которые отличаются для определенных подклассов ситуаций. Это дает возможность обращаться к подобным программным модулям по общим именам (**полиморфизм**) и осуществлять повторное использование программного кода при модификации программного обеспечения. Таким образом, адаптивность объектно-ориентированных систем к изменению предметной области по сравнению с функциональным подходом значительно выше.

При объектно-ориентированном подходе изменяется и принцип проектирования ИС. Сначала выделяются классы объектов, а далее в зависимости от возможных состояний объектов (жизненного цикла объектов) определяются методы обработки (функциональные процедуры), что обеспечивает наилучшую реализацию динамического поведения информационной системы.

Для объектно-ориентированного подхода разработаны графические методы моделирования предметной области, обобщенные в языке унифицированного моделирования UML. Однако по наглядности представления модели пользователю-заказчику объектно-ориентированные модели явно уступают функциональным моделям.

При выборе методики моделирования предметной области обычно в качестве критерия выступает степень ее динамичности. Для более регламентированных задач больше подходят функциональные модели, для более адаптивных бизнес-процессов (управления рабочими потоками, реализации динамических запросов к информационным хранилищам) — объектно-ориентированные модели. Однако в рамках одной и той же ИС для различных классов задач могут требоваться различные виды моделей, описывающих одну и ту же проблемную область. В таком случае должны использоваться комбинированные модели предметной области.

6.3 Синтетическая методика

Как можно видеть из представленного обзора, каждая из рассмотренных методик позволяет решить задачу построения формального описания рабочих процедур исследуемой системы. Все методики позволяют построить модель "как есть" и "как должно быть". С другой стороны, каждая из этих методик обладает существенными недостатками. Их можно суммировать следующим образом: недостатки применения отдельной методики лежат не в области описания реальных процессов, а в неполноте методического подхода.

Функциональные методики в целом лучше дают представление о существующих функциях в организации, о методах их реализации, причем чем выше степень детализации исследуемого процесса, тем лучше они позволяют описать систему. Под лучшим описанием в данном случае понимается наименьшая ошибка при попытке по полученной модели предсказать поведение реальной системы. На уровне отдельных рабочих процедур их описание практически однозначно совпадает с фактической реализацией в потоке работ.

На уровне общего описания системы функциональные методики допускают значительную степень произвола в выборе общих интерфейсов системы, ее механизмов и т.д., то есть в определении границ системы. Хорошо описать систему на этом уровне позволяет объектный подход, основанный на понятии сценария использования. Ключевым является понятие о сценарии использования как о сеансе взаимодействия действующего лица с системой, в результате которого действующее лицо получает нечто, имеющее для него ценность. Использование критерия ценности для пользователя дает возможность отбросить не имеющие значения детали потоков работ и сосредоточиться на тех функциях системы, которые оправдывают ее существование. Однако и в этом случае задача определения границ системы, выделения внешних пользователей является сложной.

Технология потоков данных, исторически возникшая первой, легко решает проблему границ системы, поскольку позволяет за счет анализа информационных потоков выделить внешние сущности

и определить основной внутренний процесс. Однако отсутствие выделенных управляющих процессов, потоков и событийной ориентированности не позволяет предложить эту методику в качестве единственной.

Наилучшим способом преодоления недостатков рассмотренных методик является формирование синергетической методики, объединяющей различные этапы отдельных методик. При этом из каждой методики необходимо взять часть методологии, наиболее полно и формально изложенную, и обеспечить возможность обмена результатами на различных этапах применения **синергетической методики**. В бизнес-моделировании неявным образом идет формирование подобной синергетической методики.

Идея синтетической методики заключается в последовательном применении функционального и объектного подхода с учетом возможности реинжиниринга существующей ситуации.

Рассмотрим применение **синтетической методики** на примере разработки административного регламента.

При построении административных регламентов выделяются следующие стадии:

1. Определение границ системы. На этой стадии при помощи **анализа потоков данных выделяют внешние сущности** и собственно моделируемую систему.
2. Выделение сценариев использования системы. На этой стадии **при помощи критерия** полезности **строят** для каждой внешней сущности **набор сценариев использования системы**.
3. Добавление системных сценариев использования. На этой стадии определяют **сценарии, необходимые для реализации целей системы**, отличных от целей пользователей.
4. Построение диаграммы активностей по сценариям использования. На этой стадии строят **набор действий системы**, приводящих к реализации сценариев использования;

5. Функциональная **декомпозиция диаграмм активностей** как контекстных диаграмм методики IDEF0.
6. Формальное описание отдельных функциональных активностей в виде административного регламента (с применением различных нотаций).

Контрольные вопросы

1. Что является критерием адекватности структурной модели предметной области?

Понятность для заказчиков и разработчиков

Однозначное описание структуры предметной области

Функциональная полнота разрабатываемой ИС

2. Укажите оценочные аспекты моделирования предметной области

Стоимостные затраты на обработку данных

Надежность процессов

Время решения задач

3. На каком уровне строятся модели предметной области?

На внутреннем уровне (реализации требований)

На внешнем уровне (определении требований)

На концептуальном уровне (спецификации требований)

4. Какие основные понятия используются при создании диаграммы потоков данных?

Внешние источники и получатели данных

Хранилища, требуемые процессами для своих операций

Потоки данных

Процессы преобразования входных потоков данных в выходные

Функциональный блок

5. Какие основные понятия используются при создании функциональной диаграммы IDEF0?

Функциональный блок

Внешние источники и получатели данных

Хранилища, требуемые процессами для своих операций

Декомпозиция

Интерфейсная дуга

6. Укажите основные понятия объектно-ориентированного подхода

Наследование

Полиморфизм

Объект

Функциональный блок

Класс

Внешние источники и получатели данных

7. Укажите преимущества методики DFD

Возможность однозначно определить внешние сущности

Требование скрывания информации в спецификациях и запрет переопределения уже определенных процессов в спецификациях

Возможность проектирования сверху вниз

Необходимость искусственного ввода управляющих процессов

Отсутствие понятия времени

8. Укажите преимущества объектно-ориентированной методики моделирования

Пригодность для повторного использования

Наглядность

Унификация разработки

Естественность модели

Уменьшение риска создания сложных моделей

9. Укажите преимущества функциональной методики моделирования

Возможность постепенного развития системы

Пригодность для повторного использования

Наглядность

Набрано баллов

7 Информационное обеспечение ИС

Информационное обеспечение ИС является средством для решения следующих задач:

- однозначного и экономичного представления информации в системе (на основе кодирования объектов);
- организации процедур анализа и обработки информации с учетом характера связей между объектами (на основе классификации объектов);
- организации взаимодействия пользователей с системой (на основе экранных форм ввода-вывода данных);
- обеспечения эффективного использования информации в контуре управления деятельностью объекта автоматизации (на основе унифицированной системы документации).

Информационное обеспечение ИС включает два комплекса: внешнее информационное обеспечение (классификаторы технико-экономической информации, документы, методические инструктивные материалы) и внутримашинное информационное обеспечение (макеты/экранные формы для ввода первичных данных в ЭВМ или вывода результатной информации, структуры информационной базы: входных, выходных файлов, базы данных).

К информационному обеспечению предъявляются следующие общие требования:

- информационное обеспечение должно быть достаточным для поддержания всех автоматизируемых функций объекта;
- для кодирования информации должны использоваться принятые у заказчика классификаторы;
- для кодирования входной и выходной информации, которая используется на высшем уровне управления, должны быть использованы классификаторы этого уровня;
- должна быть обеспечена совместимость с информационным обеспечением систем, взаимодействующих с разрабатываемой системой;
- формы документов должны отвечать требованиям корпоративных стандартов заказчика (или унифицированной системы документации);
- структура документов и экранных форм должна соответствовать характеристиками терминалов на рабочих местах конечных пользователей;
- графики формирования и содержание информационных сообщений, а также используемые аббревиатуры должны быть общеприняты в этой предметной области и согласованы с заказчиком;
- в ИС должны быть предусмотрены средства контроля входной и результатной информации, обновления данных в информационных массивах, контроля целостности информационной базы, защиты от несанкционированного доступа.

Информационное обеспечение ИС можно определить как совокупность единой системы классификации, унифицированной системы документации и информационной базы.

7.1 Внемашинное информационное обеспечение

Основные понятия классификации технико-экономической информации

Для того чтобы обеспечить эффективный поиск, обработку на ЭВМ и передачу по каналам связи технико-экономической информации, ее необходимо представить в цифровом виде. С этой целью ее нужно сначала упорядочить (классифицировать), а затем формализовать (закодировать) с использованием классификатора.

Классификация – это разделение множества объектов на подмножества по их сходству или различию в соответствии с принятыми методами. Классификация фиксирует закономерные связи между классами объектов. Под объектом понимается любой предмет, процесс, явление материального или нематериального свойства. Система классификации позволяет сгруппировать объекты и выделить определенные классы, которые будут характеризоваться рядом общих свойств. Таким образом, совокупность правил распределения объектов множества на подмножества называется системой классификации.

Свойство или характеристика объекта классификации, которое позволяет установить его сходство или различие с другими объектами классификации, называется **признаком** классификации. Например, признак «роль предприятия-партнера в отношении деятельности объекта автоматизации» позволяет разделить все предприятия на две группы (на два подмножества): «поставщики» и «потребители». Множество или подмножество, объединяющее часть объектов классификации по одному или нескольким признакам, носит название **классификационной группировки**.

Классификатор — это документ, с помощью которого осуществляется формализованное описание информации в ИС, содержащей наименования объектов, наименования классификационных группировок и их кодовые обозначения.

По сфере действия выделяют следующие виды классификаторов: международные, общегосудар-

ственные (общесистемные), отраслевые и локальные классификаторы.

Международные классификаторы входят в состав Системы международных экономических стандартов (СМЭС) и обязательны для передачи информации между организациями разных стран мирового сообщества.

Общегосударственные (общесистемные) классификаторы, обязательны для организации процессов передачи и обработки информации между экономическими системами государственного уровня внутри страны.

Отраслевые классификаторы используют для выполнения процедур обработки информации и передачи ее между организациями внутри отрасли.

Локальные классификаторы используют в пределах отдельных предприятий.

Каждая система классификации характеризуется следующими свойствами:

- гибкостью системы;
- емкостью системы;
- степенью заполненности системы.

Гибкость системы — это способность допускать включение новых признаков, объектов без разрушения структуры классификатора. Необходимая гибкость определяется временем жизни системы.

Емкость системы — это наибольшее количество классификационных группировок, допускаемое в данной системе классификации.

Степень заполненности системы определяется как частное от деления фактического количества группировок на величину емкости системы.

В настоящее время чаще всего применяются два типа систем классификации: иерархическая и многоаспектная.

ционной группировки, образованной по одному признаку, на множество классификационных группировок по нижестоящему (подчиненному) признаку.

Таким образом, классификационные схемы, построенные на основе иерархического принципа, имеют неограниченную емкость, величина которой зависит от глубины классификации (числа ступеней деления) и количества объектов классификации, которое можно расположить на каждой ступени. Количество же объектов на каждой ступени классификации определяется основанием кода, то есть числом знаков в выбранном алфавите кода. (Например, если алфавит – двузначные десятичные цифры, то можно на одном уровне разместить 100 объектов). Выбор необходимой глубины классификации и структуры кода зависит от характера объектов классификации и характера задач, для решения которых предназначен классификатор.

При построении иерархической системы классификации сначала выделяется некоторое множество объектов, подлежащее классифицированию, для которого определяются полное множество признаков классификации и их соподчиненность друг другу, затем производится разбиение исходного множества объектов на классификационные группировки на каждой ступени классификации.

К положительным сторонам данной системы следует отнести логичность, простоту ее построения и удобство логической и арифметической обработки.

Серьезным недостатком иерархического метода классификации является жесткость классификационной схемы. Она обусловлена заранее установленным выбором признаков классификации и порядком их использования по ступеням классификации. Это ведет к тому, что при изменении состава объектов классификации, их характеристик или характера решаемых при помощи классификатора задач требуется коренная переработка классификационной схемы. Гибкость этой системы обеспечивается только за счет ввода большой избыточности в ветвях, что приводит к слабой заполненности структуры классификатора. Поэтому при разработке классификаторов следует учитывать, что иерархический метод классификации более предпочтителен для объектов с относительно стабильными признаками и для решения стабильного комплекса задач.

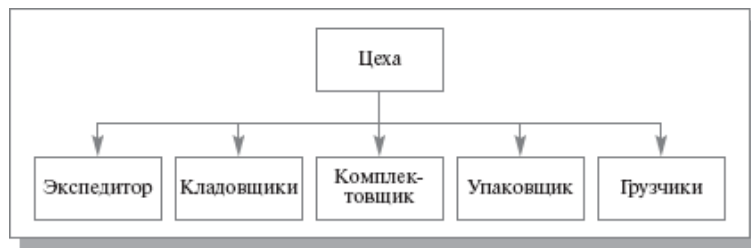


Рис. 7.2: Организационная структура подразделения предприятия-цеха отгрузки

Примеры применения иерархической классификации объектов в корпоративной ИС приведены на рис. 7.2 и 7.3. Использование приведенных моделей позволяет выполнить кодирование информации о соответствующих объектах, а также использовать процедуры обобщения при обработке данных (при анализе затрат на заработную плату — по принадлежности работника к определенной службе, при анализе затрат на производство — по группам материалов: по металлу, по покупным комплектующим и пр.).

Недостатки, отмеченные в иерархической системе, отсутствуют в других системах, которые относятся к классу многоаспектных систем классификации.

Аспект — точка зрения на объект классификации, который характеризуется одним или несколькими признаками. **Многоаспектная система** — это система классификации, которая использует параллельно несколько независимых признаков (аспектов) в качестве основания классификации. Существуют два типа многоаспектных систем: фасетная и дескрипторная. **Фасет** — это аспект классификации, который используется для образования независимых классификационных группировок. **Дескриптор** — ключевое слово, определяющее некоторое понятие, которое формирует описание объекта и дает принадлежность этого объекта к классу, группе и т.д.

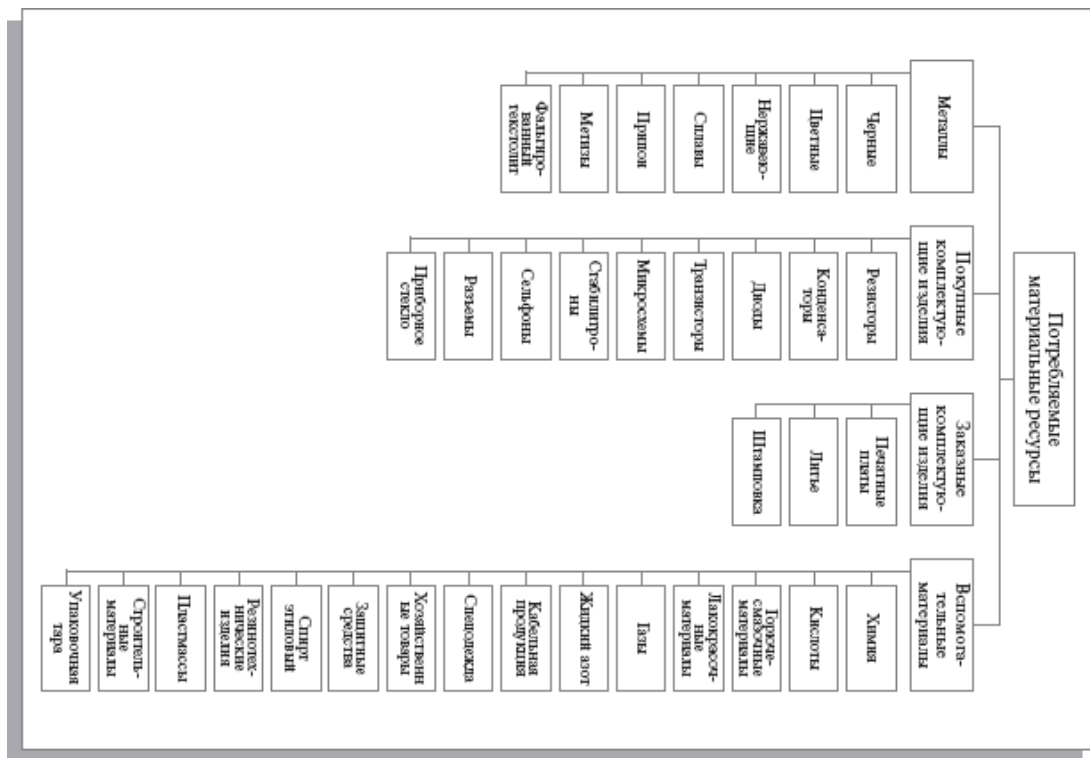


Рис. 7.3: Классификатор материальных ресурсов для обеспечения производства

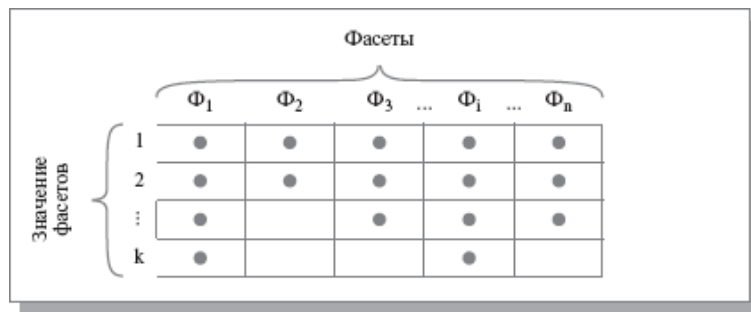


Рис. 7.4: Схема признаков фасетной классификации

Под фасетным методом классификации понимается «параллельное разделение множества объектов на независимые классификационные группировки» [7][22]. При этом методе классификации заранее жесткой классификационной схемы и конечных группировок не создается. Разрабатывается лишь система таблиц признаков объектов классификации, называемых фасетами. При необходимости создания классификационной группировки для решения конкретной задачи осуществляется выборка необходимых признаков из фасетов и их объединение в определенной последовательности. Общий вид фасетной классификационной схемы представлен на рис. 7.4.

Внутри фасета значения признаков могут просто перечисляться по некоторому порядку или образовывать сложную иерархическую структуру, если существует соподчиненность выделенных признаков.

К преимуществам данной системы следует отнести большую емкость системы и высокую степень гибкости, поскольку при необходимости можно вводить дополнительные фасеты и изменять их место в формуле. При изменении характера задач или характеристик объектов классификации

разрабатываются новые фасеты или дополняются новыми признаками уже существующие фасеты без коренной перестройки структуры всего классификатора.

К недостаткам, характерным для данной системы, можно отнести сложность структуры и низкую степень заполненности системы.

В современных классификационных схемах часто одновременно используются оба метода классификации. Это снижает влияние недостатков методов классификации и расширяет возможность использования классификаторов в информационном обеспечении управления.

В качестве примера использования комбинированных схем классификации в корпоративных ИС можно привести следующую модель описания продукции предприятия.

Правила классификации продукции

Принята классификация выпускаемой продукции по следующему ряду уровней (Иерархическая классификация):

- семейство продуктов;
- группа продуктов;
- серия продуктов.

Однако эта система классификации не обеспечивает идентификацию любого выпускаемого изделия. Для каждой единицы продукта должны указываться следующие атрибуты (Фасеты):

- код серии продукта;
- конфигурационные параметры;

- свойства.

Код серии продукта – алфавитно-цифровой код, однозначно идентифицирующий отдельный продукт. Конфигурационные параметры – свойства, значения которых могут быть различными в зависимости от потребностей пользователей. Свойства – predetermined характеристики отдельных продуктов, которые не могут меняться для одного и того же продукта.

Рассмотренные выше системы классификации хорошо приспособлены для организации поиска с целью последующей логической и арифметической обработки информации на ЭВМ, но лишь частично решают проблему содержательного поиска информации при принятии управленческих решений.

Для поиска показателей и документов по набору содержательных признаков используется информационный язык дескрипторного типа, которой характеризуется совокупностью терминов (дескрипторов) и набором отношений между терминами.

Содержание документов или показателей можно достаточно полно и точно отразить с помощью списка ключевых слов — дескрипторов. **Дескриптор** — это термин естественного языка (слово или словосочетание), используемый при описании документов или показателей, который имеет самостоятельный смысл и неделим без изменения своего значения.

Для того чтобы обеспечить точность и однозначность поиска с помощью дескрипторного языка, необходимо предварительно определить все постоянные отношения между терминами: родовидовые, отношения синонимии, омонимии и полисемии, а также ассоциативные отношения.

Все выделенные отношения явно описываются в систематическом словаре понятий — **тезаурусе**, который разрабатывается с целью проведения индексирования документов, показателей и информационных запросов.

Кодирование технико-экономической информации

Для полной формализации информации недостаточно простой классификации, поэтому проводят следующую процедуру — кодирование. **Кодирование** — это процесс присвоения условных обозначений объектам и классификационным группам по соответствующей системе кодирования. Кодирование реализует перевод информации, выраженной одной системой знаков, в другую систему, то есть перевод записи на естественном языке в запись с помощью кодов. **Система кодирования** — это совокупность правил обозначения объектов и группировок с использованием кодов. **Код** — это условное обозначение объектов или группировок в виде знака или группы знаков в соответствии с принятой системой. Код базируется на определенном алфавите (некоторое множество знаков). Число знаков этого множества называется основанием кода. Различают следующие типы алфавитов: цифровой, буквенный и смешанный.

Код характеризуется следующими параметрами:

- длиной;
- основанием кодирования;
- структурой кода, под которой понимают распределение знаков по признакам и объектам классификации;
- степенью информативности, рассчитываемой как частное от деления общего количества признаков на длину кода;
- коэффициентом избыточности, который определяется как отношение максимального количества объектов к фактическому количеству объектов.

К методам кодирования предъявляются определенные требования:

- код должен осуществлять идентификацию объекта в пределах заданного множества объектов классификации;
- желательно предусматривать использование в качестве алфавита кода десятичных цифр и букв;
- необходимо обеспечивать по возможности минимальную длину кода и достаточный резерв незанятых позиций для кодирования новых объектов без нарушения структуры классификатора.

Методы кодирования могут носить самостоятельный характер – регистрационные методы кодирования, или быть основанными на предварительной классификации объектов – классификационные методы кодирования.

Регистрационные методы кодирования бывают двух видов: порядковый и серийно-порядковый. В первом случае кодами служат числа натурального ряда. Каждый из объектов классифицируемого множества кодируется путем присвоения ему текущего порядкового номера. Данный метод кодирования обеспечивает довольно большую долговечность классификатора при незначительной избыточности кода. Этот метод обладает наибольшей простотой, использует наиболее короткие коды и лучше обеспечивает однозначность каждого объекта классификации. Кроме того, он обеспечивает наиболее простое присвоение кодов новым объектам, появляющимся в процессе ведения классификатора. Существенным недостатком порядкового метода кодирования является отсутствие в коде какой-либо конкретной информации о свойствах объекта, а также сложность машинной обработки информации при получении итогов по группе объектов классификации с одинаковыми признаками.

В серийно-порядковом методе кодирования кодами служат числа натурального ряда с закреплением отдельных серий этих чисел (интервалов натурального ряда) за объектами классификации с одинаковыми признаками. В каждой серии, кроме кодов имеющих объектов классификации, предусматривается определенное количество кодов для резерва.

Классификационные коды используют для отражения классификационных взаимосвязей объектов и группировок и применяются в основном для сложной логической обработки экономической информации. Группу классификационных систем кодирования можно разделить на две подгруппы в зависимости от того, какую систему классификации используют для упорядочения объектов: системы последовательного кодирования и параллельного кодирования.

Последовательные системы кодирования характеризуются тем, что они базируются на предварительной классификации по иерархической системе. Код объекта классификации образуется с использованием кодов последовательно расположенных подчиненных группировок, полученных при иерархическом методе кодирования. В этом случае код нижестоящей группировки образуется путем добавления соответствующего количества разрядов к коду вышестоящей группировки.

Параллельные системы кодирования характеризуются тем, что они строятся на основе использования фасетной системы классификации и коды группировок по фасетам формируются независимо друг от друга.

В параллельной системе кодирования возможны два варианта записи кодов объекта:

1. Каждый фасет и признак внутри фасета имеют свои коды, которые включаются в состав кода объекта. Такой способ записи удобно применять тогда, когда объекты характеризуются неодинаковым набором признаков. При формировании кода какого-либо объекта берутся только необходимые признаки.
2. Для определения групп объектов выделяется фиксированный набор признаков и устанавливается стабильный порядок их следования, то есть устанавливается фасетная формула. В этом случае не надо каждый раз указывать, значение какого из признаков приведено в определенных разрядах кода объекта.

Параллельный метод кодирования имеет ряд преимуществ. К достоинствам рассматриваемого метода следует отнести гибкость структуры кода, обусловленную независимостью признаков, из кодов

которых строится код объекта классификации. Метод позволяет использовать при решении конкретных технико-экономических и социальных задач коды только тех признаков объектов, которые необходимы, что дает возможность работать в каждом отдельном случае с кодами небольшой длины. При этом методе кодирования можно осуществлять группировку объектов по любому сочетанию признаков. Параллельный метод кодирования хорошо приспособлен для машинной обработки информации. По конкретной кодовой комбинации легко узнать, набором каких характеристик обладает рассматриваемый объект. При этом из небольшого числа признаков можно образовать большое число кодовых комбинаций. Набор признаков при необходимости может легко пополняться присоединением кода нового признака. Это свойство параллельного метода кодирования особенно важно при решении технико-экономических задач, состав которых часто меняется.

Наиболее сложными вопросами, которые приходится решать при разработке классификатора, являются выбор методов классификации и кодирования и выбор системы признаков классификации. Основой классификатора должны быть наиболее существенные признаки классификации, соответствующие характеру решаемых с помощью классификатора задач. При этом данные признаки могут быть или соподчиненными, или несоподчиненными. При соподчиненных признаках классификации и стабильном комплексе задач, для решения которых предназначен классификатор, целесообразно использовать иерархический метод классификации, который представляет собой последовательное разделение множества объектов на подчиненные классификационные группировки. При несоподчиненных признаках классификации и при большой динамичности решаемых задач целесообразно использовать фасетный метод классификации.

Важным вопросом является также правильный выбор последовательности использования признаков классификации по ступеням классификации при иерархическом методе классификации. Критерием при этом является статистика запросов к классификатору. В соответствии с этим критерием на верхних ступенях классификации в классификаторе должны использоваться признаки, к которым будут наиболее частые запросы. По этой же причине на верхних ступенях классификации выбирают

наименьшее основание кода.

Понятие унифицированной системы документации

Основной компонентой немашинного информационного обеспечения ИС является система документации, применяемая в процессе управления экономическим объектом. Под документом понимается определенная совокупность сведений, используемая при решении технико-экономических задач, расположенная на материальном носителе в соответствии с установленной формой.

Система документации — это совокупность взаимосвязанных форм документов, регулярно используемых в процессе управления экономическим объектом. Отличительной особенностью системы экономической документации является большое разнообразие видов документов.

Существующие системы документации, характерные для неавтоматизированных ИС, отличаются большим количеством разных типов форм документов, большим объемом потоков документов и их запутанностью, дублированием информации в документах и работ по их обработке и, как следствие, низкой достоверностью получаемых результатов. Для того чтобы упростить систему документации, используют следующие два подхода:

- проведение унификации и стандартизации документов;
- введение безбумажной технологии, основанной на использовании электронных документов и новых информационных технологий их обработки.

Унификация документов выполняется путем введения единых форм документов. Таким образом, вводится единообразие в наименования показателей, единиц измерения и терминов, в результате чего получается унифицированная система документации.

Унифицированная система документации (УСД) — это рационально организованный комплекс взаимосвязанных документов, который отвечает единым правилам и требованиям и содержит информацию, необходимую для управления некоторым экономическим объектом. По уровням управления, они делятся на межотраслевые системы документации, отраслевые и системы документации локального уровня, т. е. обязательные для использования в рамках предприятий или организаций.

Любой тип УСД должен удовлетворять следующим **требованиям**:

- документы, входящие в состав УСД, должны разрабатываться с учетом их использования в системе взаимосвязанных ЭИС;
- УСД должна содержать полную информацию, необходимую для оптимального управления тем объектом, для которого разрабатывается эта система;
- УСД должна быть ориентирована на использование средств вычислительной техники для сбора, обработки и передачи информации;
- УСД должна обеспечить информационную совместимость ЭИС различных уровней;
- все документы, входящие в состав разрабатываемой УСД, и все реквизиты-признаки в них должны быть закодированы с использованием международных, общесистемных или локальных классификаторов.

7.2 Внутримашинное информационное обеспечение

Внутримашинное информационное обеспечение включает макеты (экранные формы) для ввода первичных данных в ЭВМ или вывода результатной информации, и структуры информационной базы: входных, выходных файлов, базы данных.

Проектирование экранных форм электронных документов

Под электронными формами документов понимается не изображение бумажного документа, а изначально электронная (безбумажная) технология работы; она предполагает появление бумажной формы только в качестве твердой копии документа.

Электронная форма документа (ЭД) — это страница с пустыми полями, оставленными для заполнения пользователем. Формы могут допускать различный тип входной информации и содержать командные кнопки, переключатели, выпадающие меню или списки для выбора.

Создание форм электронных документов требует использования специального программного обеспечения. На рис. 7.5 приведены основные типовые элементы электронного документа, использование которых предусмотрено в большинстве программ автоматизации проектирования электронных документов.

К недостаткам электронных документов можно отнести неполную юридическую проработку процесса их утверждения или подписания.

Технология обработки электронных документов требует использования специализированного программного обеспечения — программ управления документооборотом, которые зачастую встраиваются в корпоративные ИС.

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

- **создание структуры ЭД** — подготовка внешнего вида с помощью графических средств проектирования;
- **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;

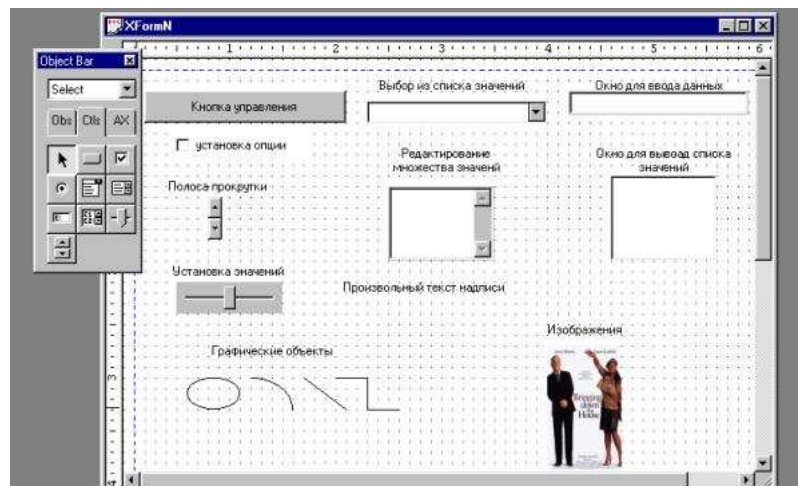


Рис. 7.5: Элементы электронного документа

- **определения перечня макетов экранных форм** — по каждой задаче проектировщик анализирует «постановку» каждой задачи, в которой приводятся перечни используемых входных документов с оперативной и постоянной информацией и документов с результатной информацией;
- **определение содержания макетов** — выполняется на основе анализа состава реквизитов первичных документов с постоянной и оперативной информацией и результатных документов.

Работа заканчивается программированием разработанных макетов экранных форм и их апробацией.

Информационная база и способы ее организации

Основной частью внутримашинного информационного обеспечения является информационная база. Информационная база (ИБ) — это совокупность данных, организованная определенным способом и хранящаяся в памяти вычислительной системы в виде файлов, с помощью которых удовлетворяются информационные потребности управленческих процессов и решаемых задач.

Все файлы ИБ можно классифицировать по следующим признакам:

- **по этапам обработки** (входные, базовые, результатные);
- **по типу носителя** (на промежуточных носителях — гибких магнитных дисках и магнитных лентах и на основных носителях — жестких магнитных дисках, магнитооптических дисках и др.);
- **по составу информации** (файлы с оперативной информацией и файлы с постоянной информацией);

- **по назначению** (по типу функциональных подсистем);
- **по типу логической организации** (файлы с линейной и иерархической структурой записи, реляционные, табличные);
- **по способу физической организации** (файлы с последовательным, индексным и прямым способом доступа).

Входные файлы создаются с первичных документов для ввода данных или обновления базовых файлов.

Файлы с результатной информацией предназначаются для вывода ее на печать или передачи по каналам связи и не подлежат долговременному хранению.

К числу базовых файлов, хранящихся в информационной базе, относят основные, рабочие, промежуточные, служебные и архивные файлы.

Основные файлы должны иметь однородную структуру записей и могут содержать записи с оперативной и условно-постоянной информацией. **Оперативные файлы** могут создаваться на базе одного или нескольких входных файлов и отражать информацию одного или нескольких первичных документов. **Файлы с условно-постоянной информацией** могут содержать справочную, расценочную, табличную и другие виды информации, изменяющейся в течение года не более чем на 40%, а следовательно, имеющие коэффициент стабильности не менее 0,6.

Файлы со справочной информацией должны отражать все характеристики элементов материального производства (материалы, сырье, основные фонды, трудовые ресурсы и т.п.). Как правило, справочники содержат информацию классификаторов и дополнительные сведения об элементах Материальной сферы, например о ценах. Нормативно-расценочные файлы должны содержать данные о нормах расхода и расценках на выполнение операций и услуг. Табличные файлы содержат сведения об экономических показателях, считающихся постоянными в течение длительного времени

(например, процент удержания, отчисления и пр.). Плановые файлы содержат плановые показатели, хранящиеся весь плановый период.

Рабочие файлы создаются для решения конкретных задач на базе основных файлов путем выборки части информации из нескольких основных файлов с целью сокращения времени обработки данных.

Промежуточные файлы отличаются от рабочих файлов тем, что они образуются в результате решения экономических задач, подвергаются хранению с целью дальнейшего использования для решения других задач. Эти файлы, так же как и рабочие файлы, при высокой частоте обращений могут быть также переведены в категорию основных файлов.

Служебные файлы предназначены для ускорения поиска информации в основных файлах и включают в себя справочники, индексные файлы и каталоги.

Архивные файлы содержат ретроспективные данные из основных файлов, которые используются для решения аналитических, например прогнозных, задач. Архивные данные могут также использоваться для восстановления информационной базы при разрушениях.

Организация хранения файлов в информационной базе должна отвечать следующим требованиям:

- полнота хранимой информации для выполнения всех функций управления и решения экономических задач;
- целостность хранимой информации, т. е. обеспечение непротиворечивости данных при вводе информации в ИБ;
- своевременность и одновременность обновления данных во всех копиях данных;
- гибкость системы, т.е. адаптируемость ИБ к изменяющимся информационным потребностям;
- реализуемость системы, обеспечивающая требуемую степень сложности структуры ИБ;

- релевантность ИБ, под которой подразумевается способность системы осуществлять поиск и выдавать информацию, точно соответствующую запросам пользователей;
- удобство языкового интерфейса, позволяющее быстро формулировать запрос к ИБ;
- разграничение прав доступа, т.е. определение для каждого пользователя доступных типов записей, полей, файлов и видов операций над ними.

Существуют следующие **способы организации ИБ**: совокупность локальных файлов, поддерживаемых функциональными пакетами прикладных программ, и интегрированная база данных, основывающаяся на использовании универсальных программных средств загрузки, хранения, поиска и ведения данных, т.е. системы управления базами данных (СУБД).

Локальные файлы вследствие специализации структуры данных под задачи обеспечивают, как правило, более быстрое время обработки данных. Однако недостатки организации локальных файлов, связанные с большим дублированием данных в информационной системе и, как следствие, несогласованностью данных в разных приложениях, а также негибкостью доступа к информации, перекрывают указанные преимущества. Поэтому организация локальных файлов может применяться только в специализированных приложениях, требующих очень высокой скорости реакции при импорте необходимых данных.

Интегрированная ИБ, т.е. база данных (БД) — это совокупность взаимосвязанных, хранящихся вместе данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для множества приложений.

Централизация управления данными с помощью СУБД обеспечивает совместимость этих данных, уменьшение синтаксической и семантической избыточности, соответствие данных реальному состоянию объекта, разделение хранения данных между пользователями и возможность подключения новых пользователей. Но централизация управления и интеграция данных приводят к проблемам

другого характера: необходимости усиления контроля вводимых данных, необходимости обеспечения соглашения между пользователями по поводу состава и структуры данных, разграничения доступа и секретности данных.

Основными способами организации БД являются создание централизованных и распределенных БД. Основным критерием выбора способа организации ИБ является достижение минимальных трудовых и стоимостных затрат на проектирование структуры ИБ, программного обеспечения системы ведения файлов, а также на перепроектирование ИБ при возникновении новых задач.

Контрольные вопросы

1. Укажите свойства системы классификации

Степень информативности

Емкость

Степень заполненности системы

Гибкость

2. Укажите характерные особенности иерархической системы классификаторов

наличие в системе неограниченного количества признаков классификации

использование параллельно нескольких независимых признаков (в качестве основания классификации)

соподчиненность признаков классификации

3. Укажите типы многоаспектных систем классификации

Фасетная

Дескрипторная

Иерархическая

4. Укажите характеристики кода системы кодирования информации

Структура кода

Основание кодирования

Емкость

Коэффициент избыточности

Длина

Степень информативности

5. Укажите, на чем базируются последовательные системы кодирования

На разрядной или комбинированной системе кодирования

На предварительной классификации по иерархической системе классификации

На использовании фасетной системы классификации

6. Укажите, на чем базируются параллельные системы кодирования

На предварительной классификации по иерархической системе классификации

На использовании фасетной системы классификации

На разрядной или комбинированной системе кодирования

7. Укажите, какие файлы относятся к числу базовых файлов, хранящихся в информационной базе

Архивные

Промежуточные

Файлы с результатной информацией

Основные

Служебные

Рабочие

8. Укажите, какие шаги обычно включает в себя процесс проектирования форм электронных документов

Определения перечня макетов экранных форм

Определение содержания формы ЭД

Апробация работы ЭД

Определение содержания макетов

Программирование разработанных макетов экранных форм и их отладка

Создание структуры ЭД

9. Укажите, какие требования должна обеспечивать организация хранения файлов в информационной базе

Целостность хранимой информации

Реализуемость системы, обеспечивающая требуемую степень сложности структуры ИБ

Удобство языкового интерфейса

Разграничение прав доступа

Гибкость, т. е. адаптируемость ИБ к изменяющимся информационным потребностям

Релевантность ИБ

Своевременность и одновременность обновления данных во всех копиях данных

Полнота хранимой информации

Набрано баллов

8 Моделирование информационного обеспечения

8.1 Моделирование данных

Одной из основных частей **информационного обеспечения** является информационная база. **Информационная база** (ИБ) представляет собой совокупность данных, организованная определенным способом и хранящаяся в памяти вычислительной системы в виде файлов, с помощью которых удовлетворяются информационные потребности управленческих процессов и решаемых задач. Разработка БД выполняется с помощью моделирования данных. **Цель моделирования данных** состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных. Наиболее распространенным **средством моделирования данных** являются **диаграммы “сущность-связь”** (ERD). С помощью ERD осуществляется детализация накопителей данных DFD – диаграммы, а также документируются информационные аспекты бизнес-системы, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их связей с другими объектами (отношений).

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 IDEF1

Наиболее распространенными методами для построения ERD-диаграмм являются метод Баркера и метод IDEF1.

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

Метод IDEF1 основан на подходе Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. На основе совершенствования метода IDEF1 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

| | |
|------------------------------|----------------------|
| Имя сущности/ Номер сущности | Служащий/44 |
| <input type="text"/> | <input type="text"/> |

Рис. 8.1: Независимые от идентификации сущности

| | |
|------------------------------|----------------------|
| Имя сущности/ Номер сущности | Проектное задание/56 |
| <input type="text"/> | <input type="text"/> |

Рис. 8.2: Зависимые от идентификации сущности

другой сущности (рис. 8.1, 8.2).

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;

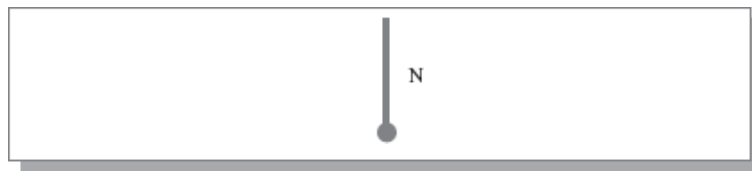


Рис. 8.3: Графическое изображение мощности связи

- 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 (рис. 8.3). 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 .

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 (рис. 8.4). 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 (рис. 8.4).

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 (рис. 8.4).

8.2 Создание логической модели данных

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).



Рис. 8.4: Неидентифицирующая связь

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.

Сущности и атрибуты

Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности — строка в таблице, а атрибуту — колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить **как объект, событие или концепцию, информация о которых должна сохраняться**. сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить "технических" наименований и быть достаточно важными для того, чтобы их моделировать. Именованная сущность в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра. Примером может быть сущности Заказчик (но не Заказчики!) с атрибутами Номер заказчика, Фамилия заказчика и Адрес заказчика. На уровне физической модели ей может соответствовать таблица Customer с колонками Customer_number, Customer_name и Customer_address. Каждая сущность должна быть полностью определена с помощью текстового описания. Для внесения дополнительных комментариев и определений к сущности служат свойства, определенные пользователем (UDP).

Как было указано выше, каждый атрибут хранит информацию об **определенном свойстве** сущности, а каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения атрибутов. Например, создание в сущности

Сотрудник атрибута Телефоны сотрудника противоречит требованиям нормализации, поскольку атрибут должен быть атомарным, т. е. не содержать множественных значений. Согласно синтаксису IDEFIX имя атрибута должно быть уникально в рамках модели (а не только в рамках сущности!).

Каждый атрибут должен быть определен, при этом следует избегать циклических определений, например, когда термин 1 определяется через термин 2, термин 2 — через термин 3, а термин 3 в свою очередь — через термин 1. Часто приходится создавать производные атрибуты, т. е. атрибуты, значение которых можно вычислить из других атрибутов. Примером производного атрибута может служить Возраст сотрудника, который может быть вычислен из атрибута Дата рождения сотрудника. Такой атрибут может привести к конфликтам; действительно, если вовремя не обновить значение атрибута Возраст сотрудника, он может противоречить значению атрибута Дата рождения сотрудника. Производные атрибуты — ошибка нормализации, однако их вводят для повышения производительности системы, чтобы не проводить вычисления, которые на практике могут быть сложными.

Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы. По умолчанию имя связи на диаграмме не показывается. На логическом уровне можно установить идентифицирующую связь "один-ко-многим" связь "многие-ко-многим" и неидентифицирующую связь "один-ко-многим".

В IDEFIX различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами. Экземпляр зависимой сущности опреде-

ляется только через отношение к родительской сущности. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ — FK.

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Идентифицирующая связь показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи, неидентифицирующая – пунктирной.

Мощность связей (Cardinality) — служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Различают четыре типа сущности:

- общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности; не помечается каким-либо символом;
- символом P помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности (исключено нулевое значение);
- символом Z помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности (исключены множественные значения);
- цифрой помечается случай точного соответствия, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.



Рис. 8.5: Пример характеристической сущности “Хобби”

Имя связи (Verb Phrase) — фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи "один-ко-многим идентифицирующей или неидентифицирующей, достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (Parent-to-Child). Для связи многие-ко-многим следует указывать имена как Parent-to-Child, так и Child-to-Parent.

Типы сущностей и иерархия наследования

Как было указано выше, связи определяют, является ли сущность независимой или зависимой. Различают несколько типов зависимых сущностей.

Характеристическая — зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности (рис. 8.5).

Ассоциативная — сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.

Именующая — частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

Категориальная — дочерняя сущность в иерархии наследования.

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, в организации работают служащие, занятые полный рабочий день (постоянные служащие), и совместители. Из их общих свойств

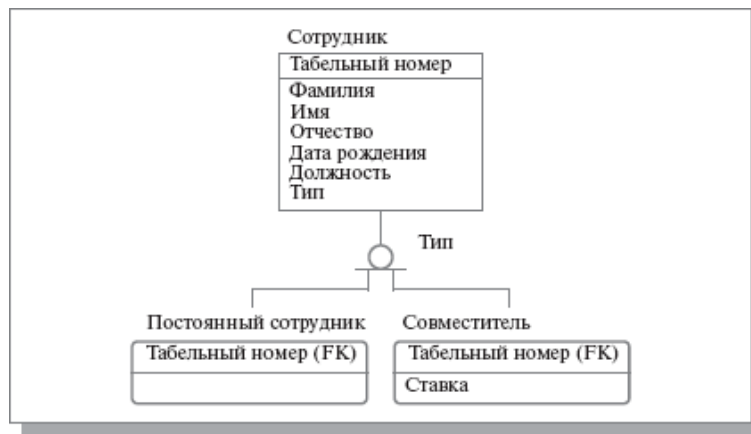


Рис. 8.6: Иерархия наследования. Неполная категория

можно сформировать обобщенную сущность (родовой предок) Сотрудник (рис. 8.6), чтобы представить информацию, общую для всех типов служащих. Специфическая для каждого типа информация может быть расположена в категориальных сущностях (потомках) Постоянный сотрудник и Совместитель.

Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи (например, если бы Постоянный сотрудник и Совместитель имели сходную по смыслу связь "работает в" с сущностью Организация), либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать дискриминатор — атрибут родového предка, который показывает, как отличить одну категориальную сущность от другой (атрибут Тип на рис. 8.6).



Рис. 8.7: Иерархия наследования. Полная категория

Иерархии категорий делятся на два типа — полные и неполные. В полной категории одному экземпляру родового предка (сущность Служащий, рис. 8.7) обязательно соответствует экземпляр в каком-либо потомке, т. е. в примере служащий обязательно является либо совместителем, либо консультантом, либо постоянным сотрудником.

Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной. На рис. 8.6 показана неполная категория — сотрудник может быть не только постоянным или совместителем, но и консультантом, однако сущность Консультант еще не внесена в иерархию наследования.

Ключи

Как было сказано выше, каждый экземпляр сущности должен быть уникален и должен отличаться от других атрибутов.

Первичный ключ (primary key) — это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности. атрибуты первичного ключа на диаграмме не требуют специального обозначения — это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии (см., например, рис. 8.7).

В одной сущности могут оказаться несколько атрибутов или наборов атрибутов, претендующих на роль первичного ключа. Такие претенденты называются потенциальными ключами (candidate key).

Ключи могут быть сложными, т. е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения — это список атрибутов, расположенных выше горизонтальной линии.

Рассмотрим кандидатов на роль первичного ключа сущности Сотрудник (рис. 8.8).

Здесь можно выделить следующие потенциальные ключи:

1. Табельный номер;
2. Номер паспорта;
3. Фамилия + Имя + Отчество.

Для того чтобы стать первичным, потенциальный ключ должен удовлетворять ряду требований:

Уникальность. Два экземпляра не должны иметь одинаковых значений возможного ключа. потенциальный ключ № 3 (Фамилия + Имя + Отчество) является плохим кандидатом, поскольку в организации могут работать полные тезки.

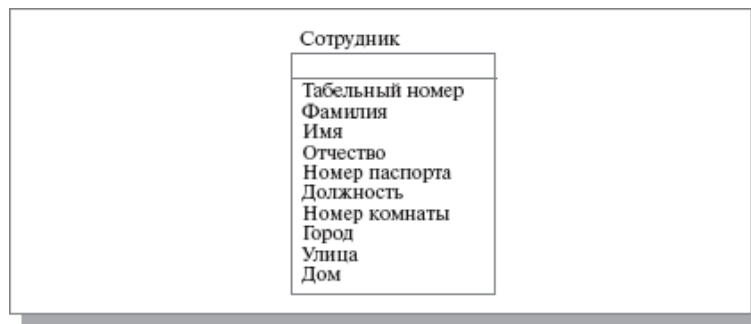


Рис. 8.8: Определение первичного ключа для сущности “Сотрудник”

Компактность. Сложный возможный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности. Для обеспечения уникальности ключа № 3 дополним его атрибутами Дата рождения и Цвет волос. Если бизнес-правила говорят, что сочетания атрибутов Фамилия + Имя + Отчество + Дата рождения достаточно для однозначной идентификации сотрудника, то Цвет волос оказывается лишним, т. е. ключ Фамилия + Имя + Отчество + Дата рождения + Цвет волос не является компактным.

При выборе первичного ключа предпочтение должно отдаваться более простым ключам, т. е. ключам, содержащим меньшее количество атрибутов. В приведенном примере ключи № 1 и 2 предпочтительней ключа № 3.

Атрибуты ключа не должны содержать нулевых значений. Значение атрибутов ключа не должно меняться в течение всего времени существования экземпляра сущности. Сотрудница организации может выйти замуж и сменить как фамилию, так и паспорт. Поэтому ключи № 2 и 3 не подходят на роль первичного ключа.

Каждая сущность должна иметь по крайней мере один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные — альтернативными ключами.

Альтернативный ключ (Alternate Key) — это потенциальный ключ, не ставший первичным.

Нормализация данных

Нормализация данных — **процесс проверки и реорганизации сущностей** и атрибутов **с целью удовлетворения требований к реляционной модели данных**. Нормализация позволяет быть уверенным, что каждый атрибут определен для своей сущности, а также значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к нормальным формам — формализованным требованиям к организации данных. Известны шесть нормальных форм.

На практике обычно ограничиваются приведением данных к третьей нормальной форме. Для углубленного изучения нормализации рекомендуется книга К. Дж. Дейта "Введение в системы баз данных"(Киев; М.: Диалектика, 1998).

ERwin не содержит полного алгоритма нормализации и не может проводить нормализацию автоматически, однако его возможности облегчают создание нормализованной модели данных. Запрет на присвоение неуникальных имен атрибутов в рамках модели (при соответствующей установке опции Unique Name) облегчает соблюдение правила "один факт — в одном месте". Имена ролей атрибутов внешних ключей и унификация атрибутов также облегчают построение нормализованной модели.

Домены

Домен можно определить как совокупность значений, из которых берутся значения атрибутов. Каждый атрибут может быть определен только на одном домене, но на каждом домене может быть определено множество атрибутов. В понятие домена входит не только тип данных, но и область значений данных. Например, можно определить домен "Возраст" как положительное целое число и определить атрибут Возраст сотрудника как принадлежащий этому домену.

В ERwin домен может быть определен только один раз и использоваться как в логической, так и в физической модели.

Домены позволяют облегчить работу с данными как разработчикам на этапе проектирования, так и администраторам БД на этапе эксплуатации системы. На логическом уровне домены можно описать без конкретных физических свойств. На физическом уровне они автоматически получают специфические свойства, которые можно изменить вручную. Так, домен "Возраст" может иметь на логическом уровне тип Number, на физическом уровне колонкам домена будет присвоен тип INTEGER.

Каждый домен может быть описан, снабжен комментарием или свойством, определенным пользователем (UDP).

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.

По умолчанию ERwin генерирует имена таблиц и индексов по шаблону на основе имен соответствующих сущностей и ключей логической модели, которые в дальнейшем могут быть откорректированы вручную. Имена таблиц и колонок будут сгенерированы по умолчанию на основе имен сущностей и атрибутов логической модели.

Правила валидации и значения по умолчанию

ERwin поддерживает правила валидации для колонок, а также значение, присваиваемое колонкам по умолчанию.

Правило валидации задает список допустимых значений для конкретной колонки и/или правила проверки допустимых значений. В список допустимых значений можно вносить новые значения. ERwin позволяет сгенерировать правила валидации соответственно синтаксису выбранной СУБД с учетом границ диапазона или списка значений.

Значение по умолчанию – значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных. С каждой колонкой или доменом можно связать значение по умолчанию. Список значений можно редактировать.

После создания правила валидации и значения по умолчанию их можно присвоить одной или

нескольким колонкам или доменами.

Индексы

В БД данные обычно хранятся в том порядке, в котором их ввели в таблицу. Многие реляционные СУБД имеют страничную организацию, при которой таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно. Такой способ позволяет быстро вводить новые данные, но затрудняет поиск данных.

Чтобы решить проблему поиска, СУБД используют объекты, называемые индексами. Индекс содержит отсортированную по колонке или нескольким колонкам информацию и указывает на строки, в которых хранится конкретное значение колонки. Поскольку значения в индексе хранятся в определенном порядке, при поиске просматривать нужно значительно меньший объем данных, что существенно уменьшает время выполнения запроса. Индекс рекомендуется создавать для тех колонок, по которым часто производится поиск.

При генерации схемы физической БД ERwin автоматически создает индекс на основе первичного ключа каждой таблицы, а также на основе всех альтернативных ключей и внешних ключей, поскольку эти колонки наиболее часто используются для поиска данных. Можно отказаться от генерации индексов по умолчанию и создать собственные индексы. Для увеличения эффективности поиска администратор БД должен анализировать часто выполняемые запросы и на основе анализа создавать собственные индексы.

Триггеры и хранимые процедуры

Триггеры и хранимые процедуры – это именованные блоки кода SQL, которые заранее откомпилированы и хранятся на сервере для того, чтобы быстро производить обработку запросов,

валидацию данных и другие часто выполняемые функции. Хранение и выполнение кода на сервере позволяет создавать код только один раз, а не в каждом приложении, работающем с БД. Это экономит время при написании и сопровождении программ. При этом гарантируется, что целостность данных и бизнес-правила поддерживаются независимо от того, какое именно клиентское приложение обращается к данным. Триггеры и хранимые процедуры не требуется пересылать по сети из клиентского приложения, что значительно снижает сетевой трафик.

Хранимой процедурой называется именованный набор предварительно откомпилированных команд SQL, который может вызываться из клиентского приложения или из другой хранимой процедуры.

Триггером называется процедура, которая выполняется автоматически как реакция на событие. Таким событием может быть вставка, изменение или удаление строки в существующей таблице. Триггер сообщает СУБД, какие действия нужно выполнить при выполнении команд SQL INSERT, UPDATE или DELETE для обеспечения дополнительной функциональности, выполняемой на сервере.

Триггер **ссылочной целостности** – это особый вид триггера, используемый для поддержания целостности между двумя таблицами, которые связаны между собой. Если строка в одной таблице вставляется, изменяется или удаляется, то триггер ссылочной целостности сообщает СУБД, что нужно делать с теми строками в других таблицах, у которых значение внешнего ключа совпадает со значением первичного ключа вставленной строки (измененной или удаленной строки).

Для генерации триггеров ERwin использует механизм шаблонов – специальных скриптов, использующих макрокоманды. При генерации кода триггера вместо макрокоманд подставляются имена таблиц, колонок, переменные и другие фрагменты кода, соответствующие синтаксису выбранной СУБД. Шаблоны триггеров ссылочной целостности, генерируемые ERwin по умолчанию, можно изменять.

Для создания и редактирования хранимых процедур ERwin располагает специальными редакторами, аналогичными редакторам, используемым для создания триггеров. В отличие от триггера

хранимая процедура не выполняется в ответ на какое-то событие, а вызывается из другой программы, которая передает на сервер имя процедуры. Хранимая процедура более гибкая, чем триггер, поскольку может вызывать другие хранимые процедуры. Ей можно передавать параметры, и она может возвращать параметры, значения и сообщения.

8.3 Проектирование хранилищ данных

В хранилища данных помещают данные, которые редко меняются. Хранилища ориентированы на выполнение аналитических запросов, обеспечивающих поддержку принятия решений для руководителей и менеджеров. При проектировании хранилищ данных необходимо выполнять следующие требования:

- хранилище должно иметь понятную для пользователей структуру данных;
- должны быть выделены статические данные, которые модифицируются по расписанию (ежедневно, еженедельно, ежеквартально);
- должны быть упрощены требования к запросам для исключения запросов, требующих множественных утверждений SQL в традиционных реляционных СУБД;
- должна обеспечиваться поддержка сложных запросов SQL, требующих обработки миллионов записей.

Как видно из этих требований, по своей структуре реляционные СУБД существенно отличаются от хранилищ данных. Нормализация данных в реляционных СУБД приводит к созданию множества связанных между собой таблиц. Выполнение сложных запросов неизбежно приводит к объединению

многих таблиц, что значительно увеличивает время отклика. Проектирование хранилища данных подразумевает создание денормализованной структуры данных, ориентированных в первую очередь на высокую производительность при выполнении аналитических запросов. Нормализация делает модель хранилища слишком сложной, затрудняет ее понимание и снижает скорость выполнения запроса. Для эффективного проектирования хранилищ данных ERwin использует размерную модель – методологию проектирования, предназначенную специально для разработки хранилищ данных. Размерное моделирование сходно с моделированием связей и сущностей для реляционной модели, но имеет другую цель. Реляционная модель акцентируется на целостности и эффективности ввода данных. Размерная модель ориентирована в первую очередь на выполнение сложных запросов

В размерном моделировании принят стандарт модели, называемый схемой "звезда" которая обеспечивает высокую скорость выполнения запроса посредством денормализации и разделения данных. Невозможно создать универсальную структуру данных, обеспечивающую высокую скорость обработки любого запроса, поэтому схема "звезда" строится для обеспечения наивысшей производительности при выполнении самого важного запроса (или группы запросов).

Схема "звезда" обычно содержит одну большую таблицу, называемую таблицей факта, помещенную в центре. Ее окружают меньшие таблицы, называемые таблицами размерности, которые связаны с таблицей факта радиальными связями.

Для создания БД со схемой "звезда" необходимо проанализировать бизнес-правила предметной области для выяснения центрального запроса. Данные, обеспечивающие выполнение этого запроса, должны быть помещены в центральную таблицу. При проектировании хранилища важно определить источник данных, метод, которым данные извлекаются, преобразуются и фильтруются, прежде чем они импортируются в хранилище. Знания об источнике данных позволяют поддерживать регулярное обновление и проверку качества данных.

Вычисление размера БД

ERwin позволяет рассчитать приблизительный размер БД в целом, а также таблиц, индексов и других объектов через определенный период времени после начала эксплуатации ИС. Расчет строится на основе следующих параметров: начальное количество строк; максимальное количество строк; прирост количества строк в месяц. Результаты расчетов сводятся в отчет.

Прямое и обратное проектирование

Прямым проектированием называется процесс генерации физической схемы БД из логической модели. При генерации физической схемы ERwin включает триггеры ссылочной целостности, хранимые процедуры, индексы, ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД.

Обратным проектированием называется процесс генерации логической модели из физической БД. Обратное проектирование позволяет конвертировать БД из одной СУБД в другую. После создания логической модели БД путем обратного проектирования можно переключиться на другой сервер и произвести прямое проектирование.

Кроме режима прямого и обратного проектирования программа обеспечивает синхронизацию между логической моделью и системным каталогом СУБД на протяжении всего жизненного цикла создания ИС.

Контрольные вопросы

1. Укажите, к какому уровню детализации относится модель данных, основанная на ключах

Модель данных среднего уровня (более подробное представление данных)

Модель данных верхнего уровня (слабо детализирована)

Модель данных нижнего уровня (детальное представление структуры данных)

2. Укажите, к какому уровню детализации относится диаграмма сущность-связь

Модель данных нижнего уровня (детальное представление структуры данных)

Модель данных верхнего уровня (слабо детализирована)

Модель данных среднего уровня (более подробное представление данных)

3. Укажите, к какому уровню детализации относится полная атрибутивная модель

Модель данных верхнего уровня (слабо детализирована)

Модель данных среднего уровня (более подробное представление данных)

Модель данных нижнего уровня (детальное представление структуры данных)

4. Укажите, что задает правило валидации:

Список допустимых значений для конкретной колонки

Правила проверки допустимых значений

Значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных

5. Укажите базовые понятия ERD-диаграммы

Сущности
Связи
Атрибуты
Идентификатор

6. Укажите, что позволяют осуществить диаграммы ERD

Документация информационных аспектов бизнес-системы
Детализация бизнес-процессов
Детализация накопителей данных

7. Укажите, какая модель данных представляет данные в третьей нормальной форме

Полная атрибутивная модель
Диаграмма сущность – связь
Модель данных, основанная на ключах

8. Укажите, какая модель данных включает описание всех сущностей и первичных ключей

Полная атрибутивная модель
Диаграмма сущность – связь
Модель данных, основанная на ключах

9. Укажите, какие уровни отображения диаграммы имеет ERwin

Уровень сущностей
Уровень первичных ключей

Уровень определений

Уровень атрибутов

Уровень иконок

Набрано баллов

9 Унифицированный язык визуального моделирования (UML)

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 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.

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 (модулей?) («ядро»), 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.

9.1 Синтаксис и семантика основных объектов UML

Классы

Классы — это базовые элементы любой объектно-ориентированной системы. Классы представляют собой описание совокупностей однородных объектов с присущими им свойствами — атрибутами, операциями, отношениями и семантикой.

В рамках модели каждому классу присваивается уникальное имя, отличающее его от других классов. Если используется составное имя (в начале имени добавляется имя пакета, куда входит класс), то имя класса должно быть уникальным в пакете.

Атрибут — это свойство класса, которое может принимать множество значений. Множество допустимых значений атрибута образует домен. Атрибут имеет имя и отражает некоторое свойство моделируемой сущности, общее для всех объектов данного класса. Класс может иметь произвольное количество атрибутов.

Операция — реализация функции, которую можно запросить у любого объекта класса. Операция показывает, что можно сделать с объектом. Исполнение операции часто связано с обработкой и изменением значений атрибутов объекта, а также изменением состояния объекта.

На рис. 9.1 приведено графическое изображение класса «Заказ» в нотации UML.

Синтаксис UML для свойств классов (в отдельных программных средствах, например, в IBM UML Modeler, порядок записи параметров может быть иным):

**<признак видимости> <имя атрибута> : <тип данных> = <значение по умолчанию>
<признак видимости> <имя операции> <(список аргументов)>**

Видимость свойства указывает на возможность его использования другими классами. Один класс может «видеть» другой, если тот находится в области действия первого и между ними существует явное или неявное отношение. В языке UML определены три уровня видимости:

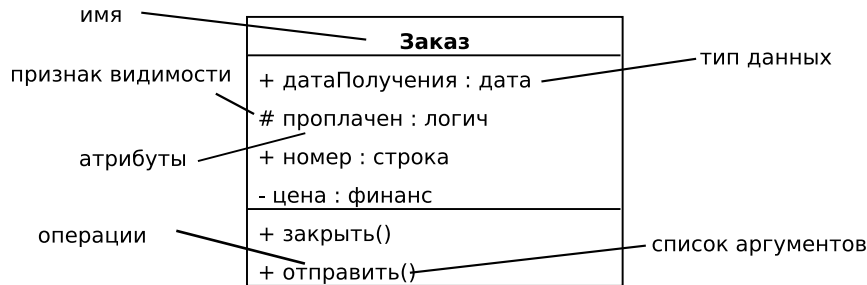


Рис. 9.1: Изображение класса в UML

- public (общий) — любой внешний класс, который «видит» данный, может пользоваться его общими свойствами. Обозначаются знаком «+» перед именем атрибута или операции;
- protected (защищенный) — только любой потомок данного класса может пользоваться его защищенными свойствами. Обозначаются знаком «#»;
- private (закрытый) — только данный класс может пользоваться этими свойствами. Обозначаются символом «-».

Еще одной важной характеристикой атрибутов и операций классов является область действия. Область действия свойства указывает, будет ли оно проявлять себя по-разному в каждом экземпляре класса, или одно и то же значение свойства будет совместно использоваться всеми экземплярами:

- instance (экземпляр) — у каждого экземпляра класса есть собственное значение данного свойства;

- classifier (классификатор) — все экземпляры совместно используют общее значение данного свойства (выделяется на диаграммах подчеркиванием).

Возможное количество экземпляров класса называется его кратностью. В UML можно определять следующие разновидности классов:

- не содержащие ни одного экземпляра — тогда класс становится служебным (Abstract);
- содержащие ровно один экземпляр (Singleton);
- содержащие заданное число экземпляров;
- содержащие произвольное число экземпляров.

Принципиальное назначение классов характеризуют стереотипы. Это, фактически, классификация объектов на высоком уровне, позволяющая определить некоторые основные свойства объекта (пример стереотипа — класс «действующее лицо»). Механизм стереотипов является также средством расширения словаря UML за счет создания на основе существующих блоков языка новых, специфичных для решения конкретной проблемы.

Диаграммы классов

Классы в UML изображаются на диаграммах классов, которые позволяют описать систему в статическом состоянии — определить типы объектов системы и различного рода статические связи между ними.

Классы отображают типы объектов системы.

Между классами возможны различные отношения, представленные на рис. 9.2:

- зависимости, которые описывают существующие между классами отношения использования;
- обобщения, связывающие обобщенные классы со специализированными;
- ассоциации, отражающие структурные отношения между объектами классов.

Зависимостью называется отношение использования, согласно которому изменение в спецификации одного элемента (например, класса «товар») может повлиять на использующий его элемент (класс «строка заказа»). Часто зависимости показывают, что один класс использует другой в качестве аргумента.

Обобщение — это отношение между общей сущностью (родителем — класс «клиент») и ее конкретным воплощением (потомком — классы «корпоративный клиент» или «частный клиент»). Объекты класса-потомка могут использоваться всюду, где встречаются объекты класса-родителя, но не наоборот. При этом он наследует свойства родителя (его атрибуты и операции). Операция потомка с той же сигнатурой, что и у(?) родителя, замещает(?) операцию родителя; это свойство называют полиморфизмом. Класс, у которого нет родителей, но есть потомки, называется корневым. Класс, у которого нет потомков, называется листовым.

Ассоциация — это отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа («клиент» может сделать «заказ»). Если между двумя классами определена ассоциация, то можно перемещаться от объектов одного класса к объектам другого. При необходимости направление навигации может задаваться стрелкой. Допускается задание ассоциаций на одном классе. В этом случае оба конца ассоциации относятся к одному и тому же классу. Это означает, что с объектом некоторого класса можно связать другие объекты из того же класса. Ассоциации может быть присвоено имя, описывающее семантику отношений. Каждая ассоциация имеет две роли, которые могут быть отражены на диаграмме (рис. 9.3). Роль ассоциации обладает свойством множественности, которое показывает, сколько соответствующих объектов может участвовать в данной связи. рис. 9.3 иллюстрирует модель формирования заказа. Каждый заказ может

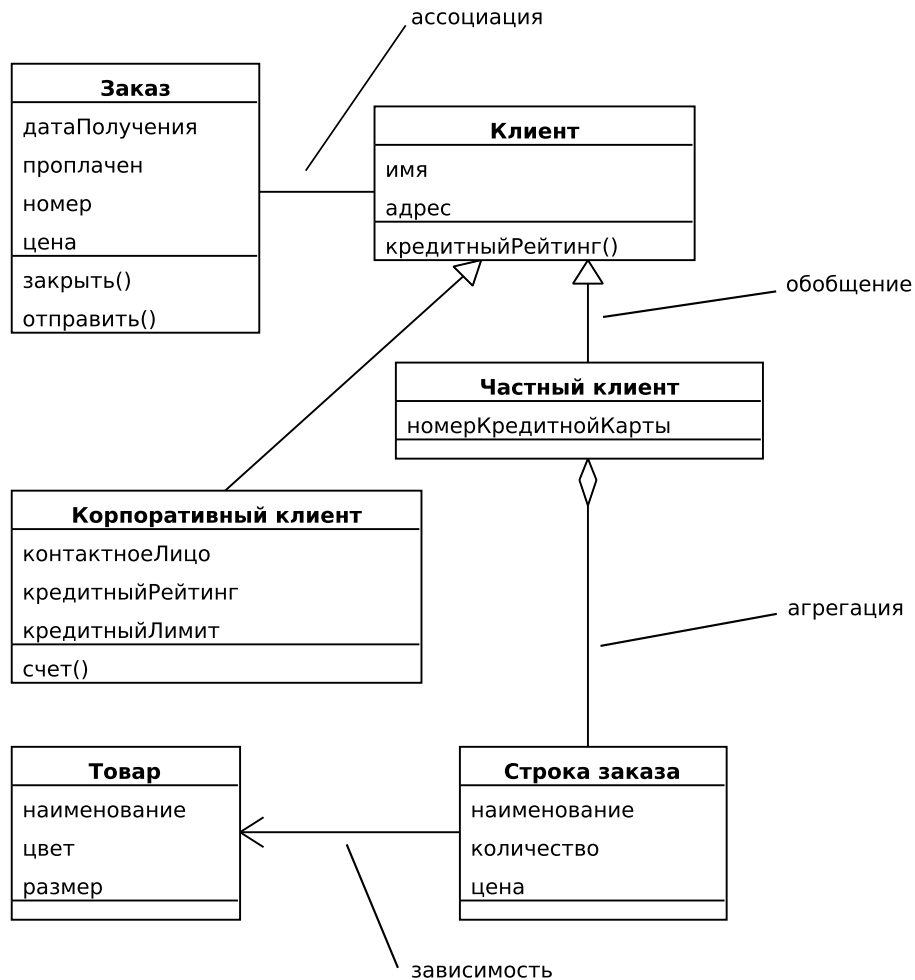


Рис. 9.2: Отображение связей между классами

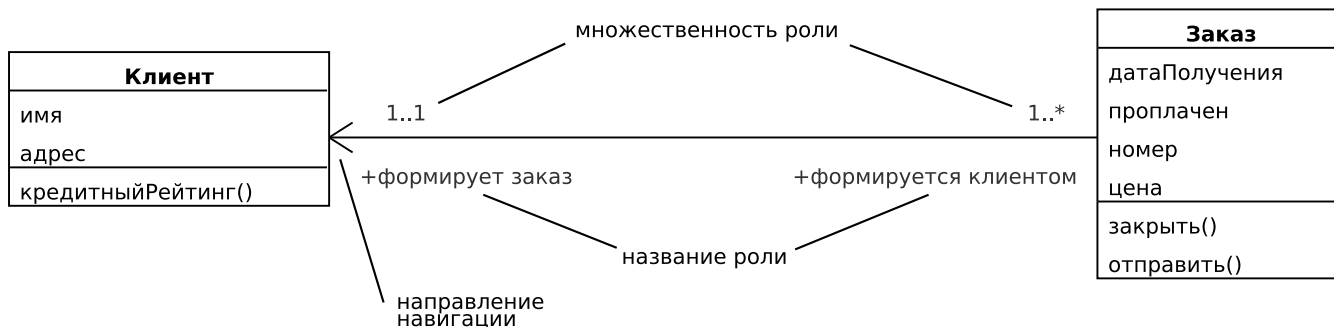


Рис. 9.3: Свойства ассоциации

быть создан единственным клиентом (множественность роли 1.1). Каждый клиент может создать один и более заказов (множественность роли 1..*). Направление навигации показывает, что каждый заказ должен быть «привязан» к определенному клиенту.

Такого рода ассоциация является простой и отражает отношение между равноправными сущностями, когда оба класса находятся на одном концептуальном уровне и ни один не является более важным, чем другой. Если приходится моделировать отношение типа «часть-целое», то используется специальный тип ассоциации — агрегирование. В такой ассоциации один из классов имеет более высокий ранг (целое — класс «заказ», рис. 9.2) и состоит из нескольких меньших по рангу классов (частей — класс «строка заказа»). В UML используется и более сильная разновидность агрегации — композиция, в которой объект-часть может принадлежать только единственному целому. В композиции жизненный цикл частей и целого совпадают, любое удаление целого обязательно захватывает и его части.

Для ассоциаций можно задавать атрибуты и операции, создавая по обычным правилам UML классы ассоциаций.

Диаграммы использования

Диаграммы использования описывают функциональность ИС, которая будет видна пользователям системы. «Каждая функциональность» изображается в виде «прецедентов использования» (use case) или просто прецедентов. Прецедент — это типичное взаимодействие пользователя с системой, которое при этом:

- описывает видимую пользователем функцию,
- может представлять различные уровни детализации,
- обеспечивает достижение конкретной цели, важной для пользователя.

Прецедент обозначается на диаграмме овалом, связанным с пользователями, которых принято называть действующими лицами (актерами, actors). Действующие лица используют систему (или используются системой) в данном прецеденте. Действующее лицо выполняет некоторую роль в данном прецеденте. На диаграмме изображается только одно действующее лицо, однако реальных пользователей, выступающих в данной роли по отношению к ИС, может быть много. Список всех прецедентов фактически определяет функциональные требования к ИС, которые лежат в основе разработки технического задания на создание системы.

На диаграммах прецедентов, кроме связей между действующими лицами и прецедентами, возможно использование еще двух видов связей между прецедентами: «использование» и «расширение» (рис. 9.4). Связь типа «расширение» применяется, когда один прецедент подобен другому, но несет несколько большую функциональную нагрузку. Ее следует применять при описании изменений в

нормальном поведении системы. Связь типа «использование» позволяет выделить некий фрагмент поведения системы и включать его в различные прецеденты без повторного описания.

На рис. 9.4 показано, что при исполнении прецедента «формирование заказа» возможно использование информации из предыдущего заказа, что позволит не вводить все необходимые данные. А при исполнении прецедентов «оценить риск сделки» и «согласовать цену» необходимо выполнить одно и то же действие — рассчитать стоимость заказа.

Динамические аспекты поведения системы отражаются приведенными ниже диаграммами.

В отличие от некоторых подходов объектного моделирования, когда и состояние, и поведение системы отображаются на диаграммах классов, UML отделяет описание поведения в диаграммы взаимодействия. В UML диаграммы классов не содержат сообщений, которые усложняют их чтение. Поток сообщений между объектами выносится на диаграммы взаимодействия. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках одного варианта использования.

Прямоугольники на диаграмме представляют различные объекты и роли, которые они имеют в системе, а линии между классами отображают отношения (или ассоциации) между ними. Сообщения обозначаются ярлыками возле стрелок, они могут иметь нумерацию и показывать возвращаемые значения.

Существуют два вида диаграмм взаимодействия: диаграммы последовательностей и кооперативные диаграммы.

Диаграммы последовательностей

Этот вид диаграмм используется для точного определения логики сценария выполнения прецедента. Диаграммы последовательностей отображают типы объектов, взаимодействующих при исполнении прецедентов, сообщения, которые они посылают друг другу, и любые возвращаемые значения, ассоциированные с этими сообщениями. Прямоугольники на вертикальных линиях показывают «время

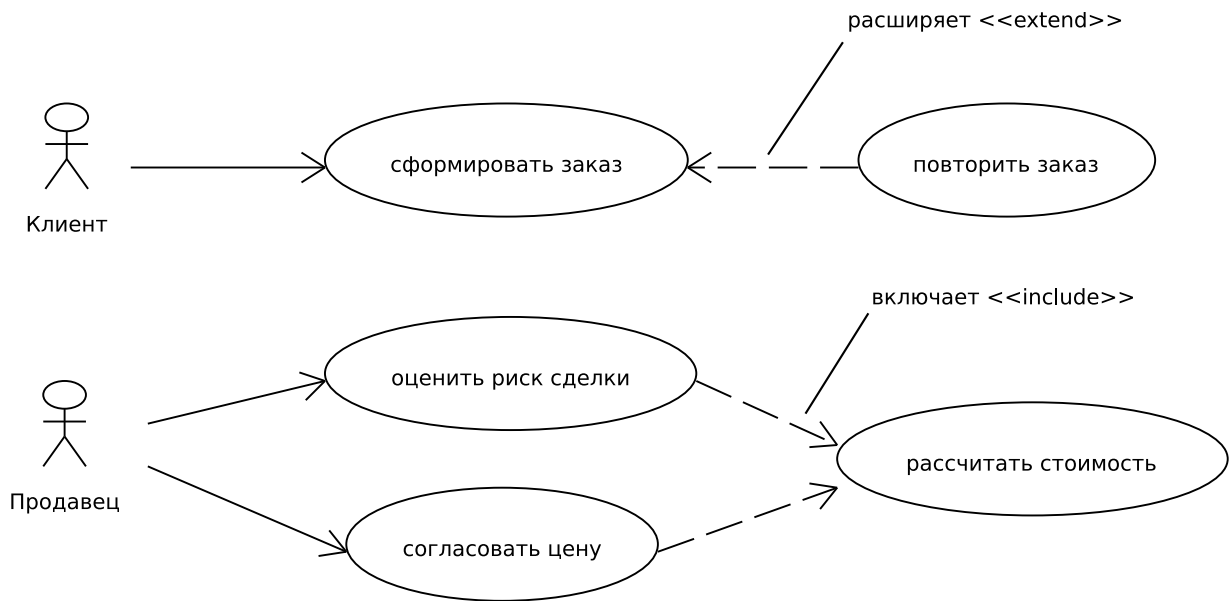


Рис. 9.4: Связи на диаграммах прецедентов

жизни» объекта. Линии со стрелками и надписями названий методов означают вызов метода у объекта.

- вводятся строки заказа;
- по каждой строке проверяется наличие товара;
- если запас достаточен — инициируется поставка;
- если запас недостаточен — инициируется дозаказ (повторный заказ).

Сообщения появляются в той последовательности, как они показаны на диаграмме — сверху вниз. Если предусматривается отправка сообщения объектом самому себе (самомоделирование), то стрелка начинается и заканчивается на одной «линии жизни».

На диаграммы может быть добавлена управляющая информация: описание условий, при которых посылается сообщение; признак многократной отправки сообщения (маркер итерации); признак возврата сообщения.

Кооперативные диаграммы

На кооперативных диаграммах объекты (или классы) показываются в виде прямоугольников, а стрелками обозначаются сообщения, которыми они обмениваются в рамках одного варианта использования. Временная последовательность сообщений отражается их нумерацией.

Диаграммы состояний

Диаграммы состояний используются для описания поведения сложных систем. Они определяют все возможные состояния, в которых может находиться объект, а также процесс смены состояний объ-

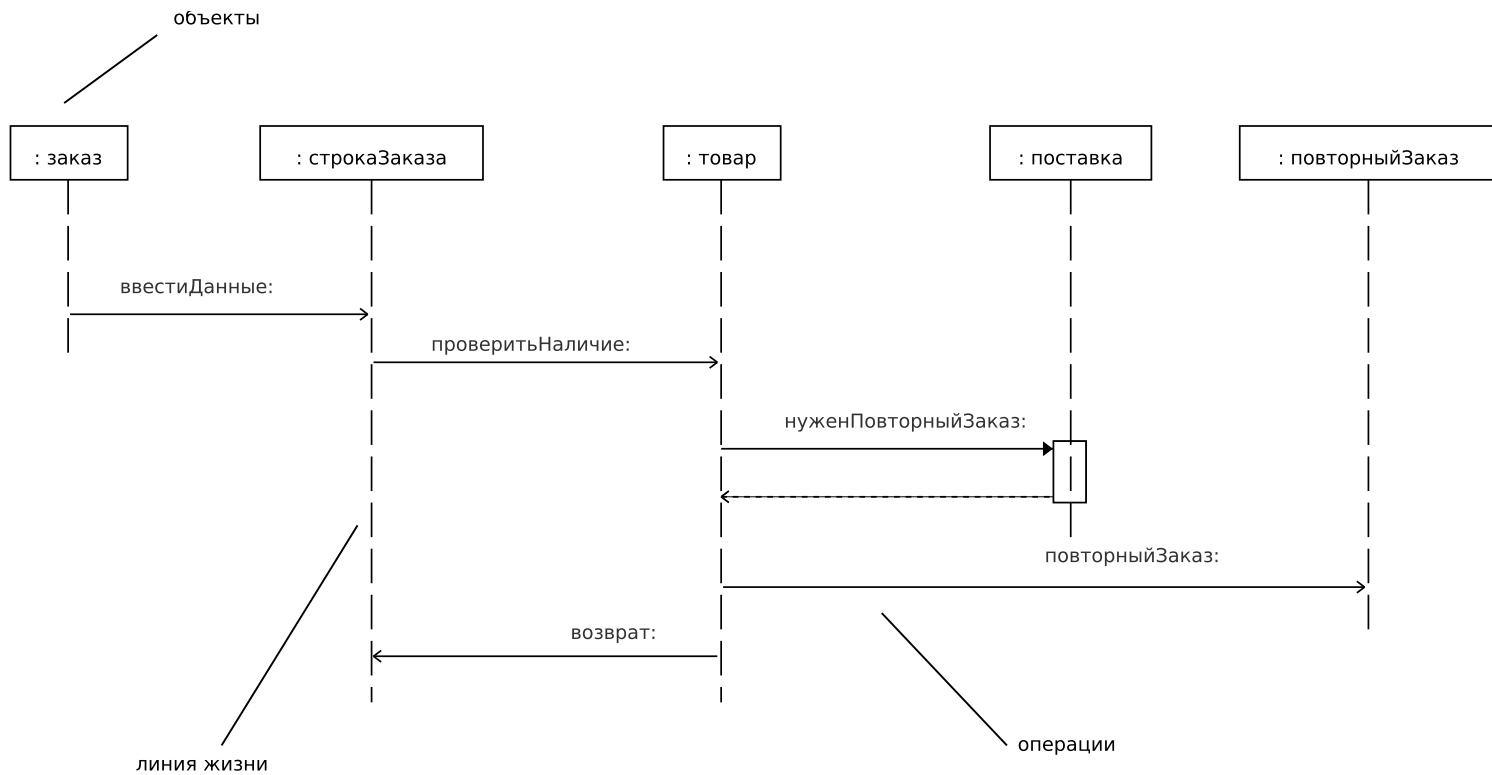


Рис. 9.5: Диаграмма последовательности обработки заказа

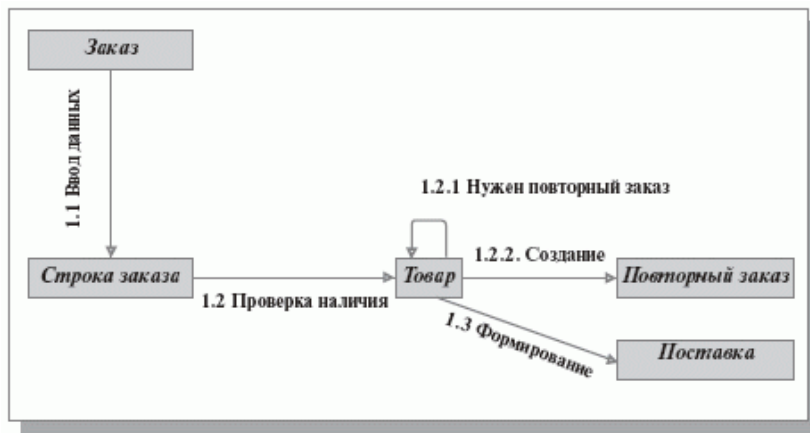


Рис. 9.6: Кооперативная диаграмма прохождения заказа

екта в результате некоторых событий. Эти диаграммы обычно используются для описания поведения одного объекта в нескольких прецедентах.

Прямоугольниками представляются состояния, через которые проходит объект во время своего поведения. Состояниям соответствуют определенные значения атрибутов объектов. Стрелки представляют переходы от одного состояния к другому, которые вызываются выполнением некоторых функций объекта. Имеется также два вида псевдо-состояний: начальное состояние, в котором находится только что созданный объект, и конечное состояние, которое объект не покидает, как только туда перешел.

Переходы имеют метки, которые синтаксически состоят из трех необязательных частей (см. рис. 9.7):

`<Событие> <[Условие]> < / Действие> .`

На диаграммах также отображаются функции, которые выполняются объектом в определенном состоянии. Синтаксис метки деятельности:

`выполнить / < деятельность > .`

Диаграммы деятельности

Диаграмма деятельности — это частный случай диаграммы состояний. На диаграмме деятельности представлены переходы потока управления от одной деятельности к другой внутри системы. Этот вид диаграмм обычно используется для описания поведения, включающего в себя множество параллельных процессов.

Основными элементами диаграмм деятельности являются (рис. 9.8):

- овалы, изображающие действия объекта;

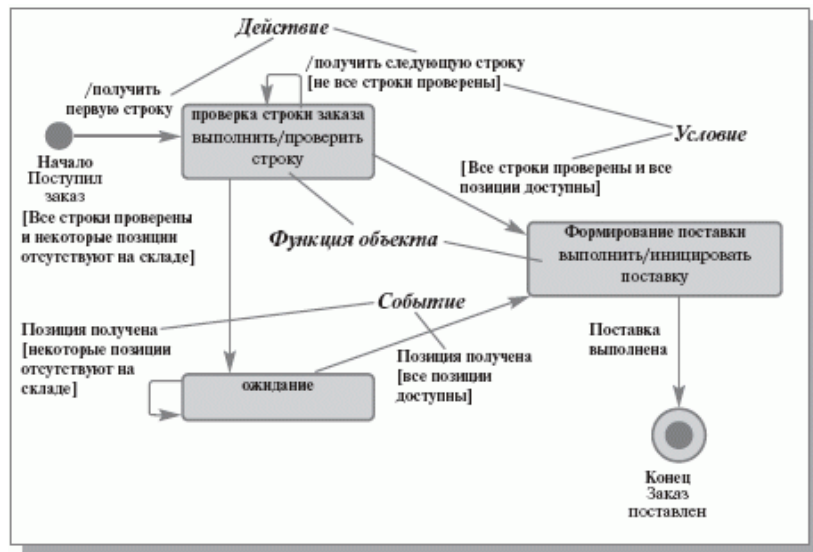


Рис. 9.7: Диаграмма состояний объекта «заказ»

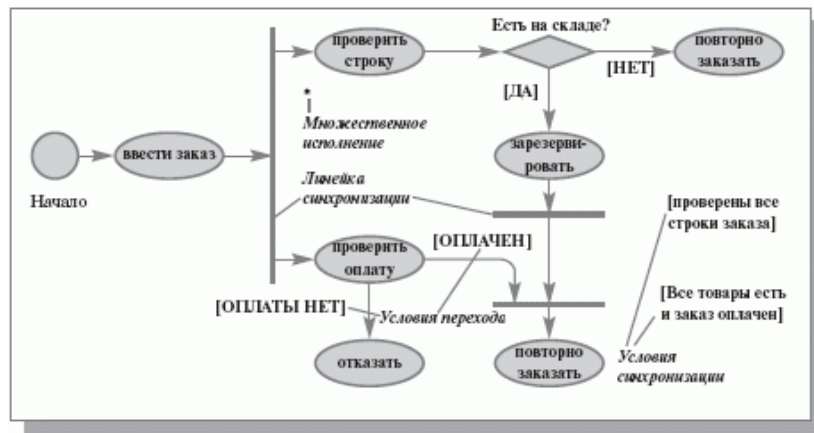


Рис. 9.8: Диаграмма деятельности — обработка заказа

- линейки синхронизации, указывающие на необходимость завершить или начать несколько действий (модель логического условия «И»);
- ромбы, отражающие принятие решений по выбору одного из маршрутов выполнения процесса (модель логического условия «ИЛИ»);
- стрелки — отражают последовательность действий, могут иметь метки условий.

На диаграмме деятельности могут быть представлены действия, соответствующие нескольким вариантам использования. На таких диаграммах появляется множество начальных точек, поскольку они отражают теперь реакцию системы на множество внешних событий. Таким образом, диаграммы деятельности позволяют получить полную картину поведения системы и легко оценивать влияние изменений в отдельных вариантах использования на конечное поведение системы.

Любая деятельность может быть подвергнута дальнейшей декомпозиции и представлена в виде отдельной диаграммы деятельности или спецификации (словесного описания).

Диаграммы компонентов

Диаграммы компонентов позволяют изобразить модель системы на физическом уровне.

Элементами диаграммы являются компоненты — физические замещаемые модули системы. Каждый компонент является полностью независимым элементом системы. Разновидностью компонентов являются узлы. Узел — это элемент реальной (физической) системы, который существует во время функционирования программного комплекса и представляет собой вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и способностью обработки. Узлы делятся на два типа:

- устройства — узлы системы, в которых данные не обрабатываются.

- процессоры — узлы системы, осуществляющие обработку данных.

Для различных типов компонентов предусмотрены соответствующие стереотипы в языке UML.

Компонентом может быть любой достаточно крупный модульный объект, такой как таблица или экстенс базы данных, подсистема, бинарный исполняемый файл, готовая к использованию система или приложение. Таким образом, диаграмму компонентов можно рассматривать как диаграмму классов в более крупном (менее детальном) масштабе. Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как классы, интерфейсы и кооперации.

Основное назначение диаграмм компонентов — разделение системы на элементы, которые имеют стабильный интерфейс и образуют единое целое. Это позволяет создать ядро системы, которое не будет меняться в ответ на изменения, происходящие на уровне подсистем.

На рис.9.9 показана упрощенная схема элементов фрагмента корпоративной системы. «Коробки» представляют собой компоненты — приложения или внутренние подсистемы. Пунктирные линии отражают зависимости между компонентами.

Каждый компонент диаграммы при необходимости документируется с помощью более детальной диаграммы компонентов, диаграммы сценариев или диаграммы классов.

Пакеты UML

Пакеты представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить диаграммы различного типа и назначения. В отличие от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки. Изображается пакет в виде папки с закладкой, содержащей, как правило, только имя и иногда — описание содержимого.

Диаграмма пакетов содержит пакеты классов и зависимости между ними. Зависимость между двумя пакетами имеет место в том случае, если изменения в определении одного элемента влекут

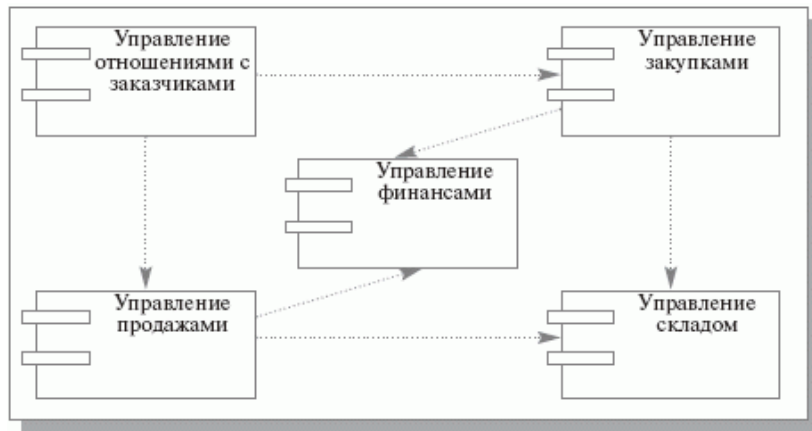


Рис. 9.9: Диаграмма компонентов фрагмента КИС

за собой изменения в другом. По отношению к пакетам можно использовать механизм обобщения (см. выше раздел «Диаграммы классов»).

Контрольные вопросы

1. Укажите основные свойства языка моделирования UML

Является основой CASE-средств нижнего уровня (lower CASE tools)

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

2. Что представляет собой класс в UML?

Описание объекта

Описание совокупности однородных объектов

Описание связи между объектами

3. Что такое «атрибут класса»?

Наименование класса

Свойство объектов класса, которое может принимать множество значений

Числовая характеристика допустимого количества объектов в классе

4. Что определяет свойство «видимость атрибута»?

Возможность отображения атрибута в экранных формах

Область действия атрибута

Возможность использования атрибута другими классами

5. Укажите возможные значения видимости свойства класса

Abstract (служебный)

Protected (защищённый)

Private (закрытый)

Singleton (единственный)

6. Укажите возможные типы отношений между классами UML

Иерархия

Ассоциация

Зависимость

Обобщения

7. Определите назначение диаграммы использования

Описывает взаимосвязи между объектами системы

Определяет последовательность действий при выполнении некоторой функции

Описывает функциональность ИС, которая будет видна пользователям системы

8. Определите назначение диаграмм последовательностей

Описывают последовательные изменения состояния системы

Используются для точного определения логики сценария выполнения прецедента

Отражают переходы потока управления от одной деятельности к другой внутри системы

9. В каких случаях целесообразно использовать диаграммы деятельности?

Для описания потока сообщений, которыми обмениваются объекты

Для описания взаимодействия пользователей с системой

Для описания поведения, включающего в себя множество параллельных процессов

Набрано баллов

10 Этапы проектирования ИС с применением UML

UML обеспечивает поддержку всех этапов жизненного цикла ИС и предоставляет для этих целей ряд графических средств – диаграмм.

На этапе создания концептуальной модели для описания бизнес-деятельности используются модели бизнес-прецедентов и диаграммы видов деятельности, для описания бизнес-объектов – модели бизнес-объектов и диаграммы последовательностей.

На этапе создания логической модели ИС описание требований к системе задается в виде модели и описания системных прецедентов, а предварительное проектирование осуществляется с использованием диаграмм классов, диаграмм последовательностей и диаграмм состояний.

На этапе создания физической модели детальное проектирование выполняется с использованием диаграмм классов, диаграмм компонентов, диаграмм развертывания.

Ниже приводятся определения и описывается назначение перечисленных диаграмм и моделей применительно к задачам проектирования ИС (в скобках приведены альтернативные названия диаграмм, используемые в современной литературе).

Диаграммы прецедентов (диаграммы вариантов использования, use case diagrams) – это обобщенная модель функционирования системы в окружающей среде.

Диаграммы видов деятельности (диаграммы деятельностей, activity diagrams) – модель бизнес-

процесса или поведения системы в рамках прецедента.

Диаграммы взаимодействия (interaction diagrams) – модель процесса обмена сообщениями между объектами, представляется в виде диаграмм последовательностей (sequence diagrams) или кооперативных диаграмм (collaboration diagrams).

Диаграммы состояний (statechart diagrams) – модель динамического поведения системы и ее компонентов при переходе из одного состояния в другое.

Диаграммы классов (class diagrams) – логическая модель базовой структуры системы, отражает статическую структуру системы и связи между ее элементами.

Диаграммы базы данных (database diagrams) – модель структуры базы данных, отображает таблицы, столбцы, ограничения и т.п.

Диаграммы компонентов (component diagrams) – модель иерархии подсистем, отражает физическое размещение баз данных, приложений и интерфейсов ИС.

Диаграммы развертывания (диаграммы размещения, deployment diagrams) – модель физической архитектуры системы, отображает аппаратную конфигурацию ИС.

На рис. 10.1 показаны отношения между различными видами диаграмм UML. Указатели стрелок можно интерпретировать как отношение "является источником входных данных для..."(например, диаграмма прецедентов является источником данных для диаграмм видов деятельности и последовательности). Приведенная схема является наглядной иллюстрацией итеративного характера разработки моделей с использованием UML.

Ниже приводятся описания последовательных этапов проектирования ИС с использованием UML.

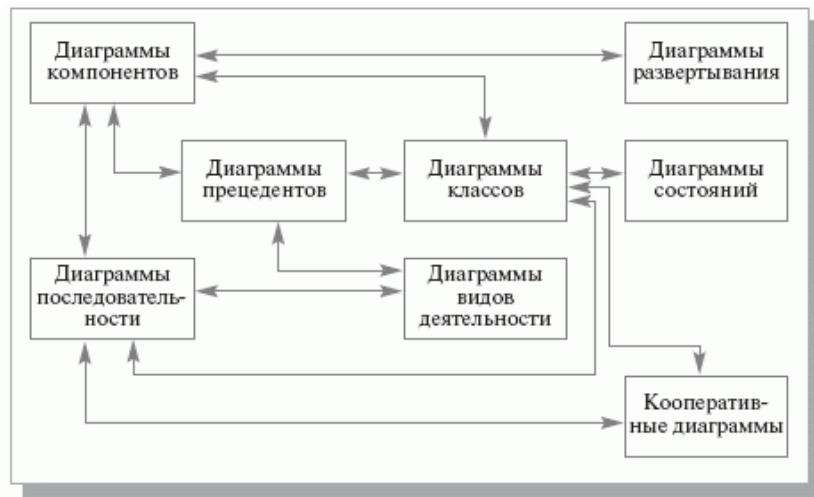


Рис. 10.1: Взаимосвязи между диаграммами UML

10.1 Разработка модели бизнес-прецедентов

Модель бизнес-прецедентов описывает бизнес-процессы с точки зрения внешнего пользователя, т.е. отражает взгляд на деятельность организации извне.

Проектирование системы начинается с изучения и моделирования бизнес-деятельности организации. На этом этапе вводится и отображается в модели ряд понятий, свойственных объектно-ориентированному подходу:

Исполнитель (Действующее лицо, Actor) – личность, организация или система, взаимодействующая с ИС; различают внешнего исполнителя (который использует или используется системой, т.е. порождает прецеденты деятельности) и внутреннего исполнителя (который обеспечивает реализацию прецедентов деятельности внутри системы). На диаграмме исполнитель представляется стилизованной фигуркой человека.

Прецедент – законченная последовательность действий, инициированная внешним объектом (личностью или системой), которая взаимодействует с ИС и получает в результате некоторое сообщение от ИС. На диаграмме представляется овалом с надписью, отражающей содержание действия.

Класс — описание совокупности однородных объектов с их атрибутами, операциями, отношениями и семантикой. На диаграмме представляется прямоугольником, содержащим описания атрибутов и операций класса.

Ассоциация – связь между двумя элементами модели. На диаграмме представляется линией.

Обобщение – связь между двумя элементами модели, когда один элемент (подкласс) является частным случаем другого элемента (суперкласса). На диаграмме представляется стрелкой.

Агрегация – отношение между элементами модели, когда один элемент является частью другого элемента (агрегата). На диаграмме представляется стрелкой с ромбовидным концом.

Для иллюстрации этапов разработки проекта использованы адаптированные материалы проекта ИС медицинского центра (рис. 10.2). Назначение ИС – автоматизация ведения и использования

клинических записей о пациентах. В настоящее время эта работа производится вручную персоналом центра. На рис. 10.2 представлена общая модель деятельности центра в виде диаграммы прецедентов. Прецедент "Обслуживание пациента" реализуется через множество других, более ограниченных прецедентов (рис. 10.3), отражающих детализацию представления функционирования центра.

Для включения в диаграмму выбранные прецеденты должны удовлетворять следующим критериям:

- прецедент должен описывать, **ЧТО** нужно делать, а не **КАК**;
- прецедент должен описывать действия с точки зрения **ИСПОЛНИТЕЛЯ**;
- прецедент должен возвращать исполнителю некоторое **СООБЩЕНИЕ**;
- последовательность действий внутри прецедента должна представлять собой одну **НЕДЕЛИМУЮ** цепочку.

Исходя из цели создания системы, для дальнейшего исследования и моделирования отбираются только те бизнес-прецеденты, которые связаны с использованием клинических записей.

Выполнение прецедента описывается с помощью диаграмм видов деятельности, которые отображают исполнителей и последовательность выполнения соответствующих бизнес-процессов (рис. 10.4).

Несмотря на то, что оказание медицинской помощи предусматривает множество разнообразных действий исполнителей, для нашей задачи существенными являются только процессы обмена информацией между этими исполнителями, и именно они отображаются в создаваемых моделях. Поэтому на диаграмме отражен процесс оценки состояния пациента на основании имеющейся в центре информации о нем.

Общее поле диаграммы деятельности делится на несколько "плавательных дорожек каждая из которых содержит описание действий одного из исполнителей. Основными элементами диаграмм



Рис. 10.2: Общая диаграмма деятельности медицинского центра по обслуживанию пациента

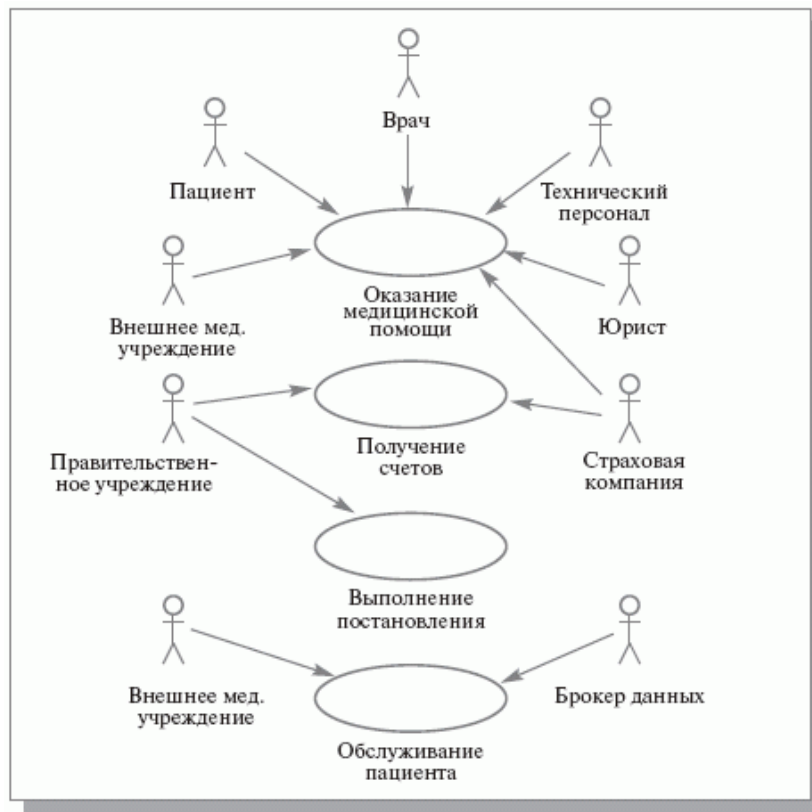


Рис. 10.3: Модель бизнес-прецедентов, составляющих обслуживание пациента

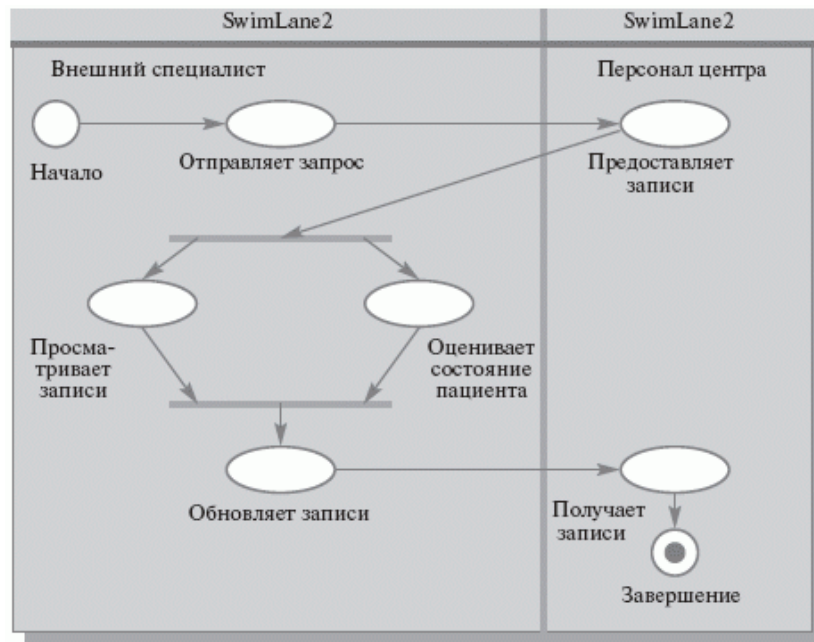


Рис. 10.4: Диаграмма видов деятельности для прецедента “Оказание медицинской помощи”

видов деятельности являются обозначения состояния ("начало" "конец"), действия (овал) и момента синхронизации действий (линейка синхронизации, на которой сходятся или разветвляются несколько стрелок).

Диаграмма подходит для описания действий как внешнего, так и внутреннего специалиста центра.

Этап завершается после разработки диаграмм видов деятельности для всех выделенных в модели бизнес-прецедентов. Естественно, на последующих этапах анализа и проектирования будут выявлены какие-то важные подробности в описании деятельности объекта автоматизации. Поэтому разработанные на данном этапе модели будут еще неоднократно корректироваться.

10.2 Разработка модели бизнес-объектов

Следующим этапом проектирования ИС является разработка модели бизнес-объектов, которая показывает выполнение бизнес-процессов организации ее внутренними исполнителями. Основными компонентами моделей бизнес-объектов являются внешние и внутренние исполнители, а также бизнес-сущности, отображающие все, что используют внутренние исполнители для реализации бизнес-процессов. Пример модели бизнес-объектов для прецедента "Ответ на запрос" приведен на рис. 10.5.

В этой диаграмме появилось новое действующее лицо – отправитель запроса. На самом деле с запросом о состоянии пациента могут обращаться в систему многие из действующих лиц: юрист, страховая компания, технический персонал и даже сам пациент. Таким образом, понятие "Отправитель запроса" служит для обобщенного представления всех этих действующих лиц при описании прецедента "Ответ на запрос" (рис. 10.6). "Отправитель запроса" становится суперклассом по отношению к обобщаемым понятиям (подклассам). Для детального описания выполнения бизнес-процессов обычно используются диаграммы последовательностей (рис. 10.7).



Рис. 10.5: Модель бизнес-объектов прецедента “Ответ на запрос”

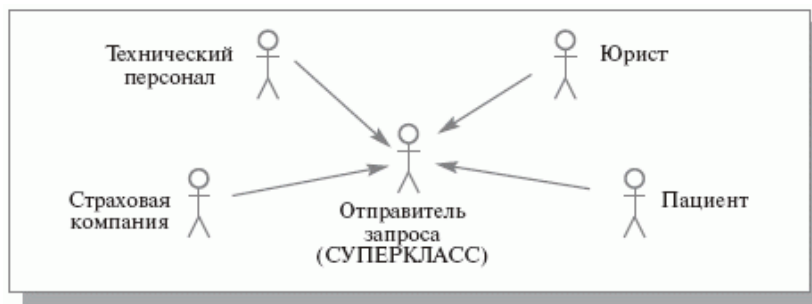


Рис. 10.6: Обобщение классов



Рис. 10.7: Диаграмма последовательностей для прецедента “Ответ на запрос”

Основными элементами диаграммы последовательностей являются обозначения объектов (прямоугольники), вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами.

Результатом этого этапа являются согласованные с заказчиком и достаточно подробные описания действий специалистов организации, внедряющей ИС, необходимые для обеспечения исполнения ее функций.

10.3 Разработка концептуальной модели данных

Затем на основе информации, выявленной на этапах бизнес-моделирования, выполняется разработка концептуальной модели данных, которые будут использоваться в разрабатываемой системе. На

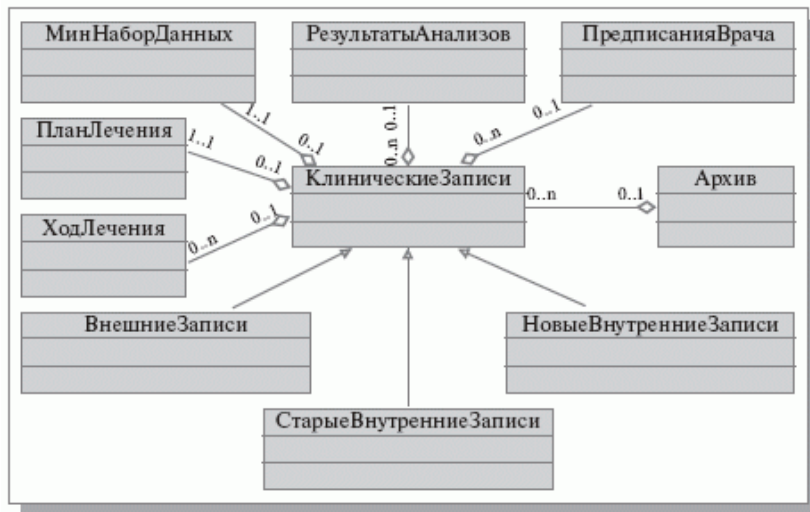


Рис. 10.8: Концептуальная модель данных

рис. 10.8 представлена в виде диаграммы классов модель данных для объекта "Клинические записи".

Модель показывает, что клинические записи включают (агрегируют) ряд блоков. При этом "минимальный набор данных" и "план лечения" могут быть включены в каждую клиническую запись в единственном экземпляре, а блоки "результаты анализов" "предписания врача" "ход лечения" могут повторяться неограниченное число раз.

Архив состоит из множества клинических записей (агрегирует клинические записи), но может

быть и пустым.

Поскольку пациент может предварительно проходить лечение в других учреждениях, или несколько раз проходить лечение в центре, появляются дополнительные разновидности (подклассы) клинических записей: внешние, старые внутренние, новые внутренние.

Этот этап завершает процедуры бизнес-моделирования и позволяет представить команде проектировщиков в едином формате ту информацию, которая будет необходима для создания системы. Разработанные диаграммы являются отправной точкой в процессах проектирования баз данных и приложений системы, обеспечивают согласованность действий бизнес-аналитиков и разработчиков в процессе дальнейшей работы над системой. Эти диаграммы, конечно же, будут претерпевать изменения в процессе последующего проектирования, однако эти изменения будут фиксироваться в формате, уже привычном для всей команды разработчиков, и будут автоматически отражаться в последующих моделях.

10.4 Разработка требований к системе

На этапе формирования требований, прежде всего, необходимо определить область действия разрабатываемой системы и получить точное представление о желаемых возможностях системы.

Основой разработки требований является модель системных прецедентов, отражающая выполнение конкретных обязанностей внутренними и внешними исполнителями с использованием ИС.

Источником данных для создания модели системных прецедентов являются разработанные на предыдущем этапе бизнес-модели. Однако при создании модели полезно предварительно составить детальные описания прецедентов, содержащие определения используемых данных и точную последовательность их выполнения. Описание осуществляется в соответствии с принятым в организации шаблоном, который обычно включает следующие разделы:

- заголовок (название прецедента, ответственный за исполнение, дата создания шаблона/внесения изменений);
- краткое описание прецедента;
- ограничения;
- предусловия (необходимое состояние системы или условия, при которых должен выполняться прецедент);
- постусловия (возможные состояния системы после выполнения прецедента);
- предположения;
- основная последовательность действий;
- альтернативные последовательности действий и условия, их иницирующие;
- точки расширения и включения прецедентов.

В процессе создания модели системных прецедентов осуществляется преобразование и перенос компонентов бизнес-моделей на новые диаграммы. Типовые преобразования по технологии Rational Unified Process приведены в таблица 10.1.

На рис. 10.9 представлена модель системных прецедентов для бизнес-прецедента "Оказание медицинской помощи". Исходя из цели создания системы, в модели системных прецедентов отражены только те действия исполнителей, которые связаны с предоставлением доступа и обновлением клинических записей.

| нес-модели | Элементы модели системных прецедентов |
|-----------------------|---------------------------------------|
| | Подсистемы |
| | Исполнители |
| | Исполнители или прецеденты |
| треними исполнителями | Прецеденты |

Описываемые моделью функции характерны только для одного вида деятельности – оказания медицинской помощи, и в основном не используются в других видах деятельности Центра. Это позволяет объединить выделенные функции в некую единую подсистему проектируемой ИС.

Внутренний исполнитель "Персонал центра"(см. рис. 10.4, рис. 10.7) и выполняемый им ручной процесс преобразован в системный прецедент "Предоставление доступа к клиническим записям".

Внешние исполнители (например, "Производитель медицинского оборудования") непосредственно взаимодействуют с проектируемой системой, т.е. превращаются в исполнителей.

В модели отражены два специальных типа связи между прецедентами (на рис. 10.9 соответствующие прецеденты выделены тенью):

- "включает"— один прецедент в процессе своего исполнения обязательно выполняет некий блок действий, составляющих другой прецедент;
- "расширяет"— когда прецеденты подобны по своим действиям, но один несет несколько большую функциональную нагрузку.

Прецедент "Проверка прав доступа"впервые появился на диаграммах и реализуется средствами разрабатываемой ИС. Поэтому для него приходится разрабатывать диаграмму последовательностей, описывающую его исполнение (рис. 10.10). В результате в проектируемой ИС появляются два новых объекта – программный модуль "Менеджер защиты"и информационный блок "Набор прав".

Таким образом, результатом разработки модели системных прецедентов является не только исчерпывающий перечень функций, которые должны быть реализованы в проектируемой системе, но и подробное описание необходимой реализации этих функций.

Анализ требований и предварительное проектирование системы.

Основные задачи этапа:

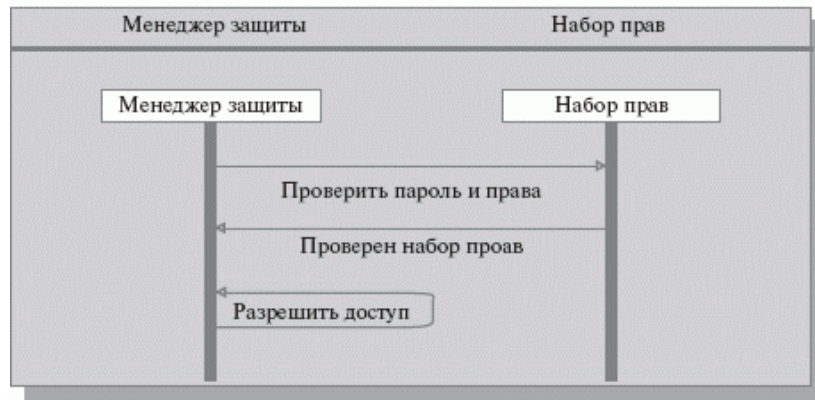


Рис. 10.10: Диаграмма последовательностей для прецедента “Проверка прав”

- определить проект системы, который будет отвечать всем бизнес-требованиям;
- разработать общий предварительный проект для всех команд разработчиков (проектировщиков баз данных, разработчиков приложений, системных архитекторов и пр.)

Основным инструментом на данном этапе являются диаграммы классов системы, которые строятся на основе разработанной модели системных прецедентов. Одновременно на этом этапе уточняются диаграммы последовательностей выполнения отдельных прецедентов, что приводит к изменениям в составе объектов и диаграммах классов. Это естественное отражение средствами UML итеративного процесса разработки системы.

Диаграммы классов системы заполняются объектами из модели системных прецедентов. Они содержат описание этих объектов в виде классов и описание взаимодействия между классами.

Диаграмма классов, описывающая процедуры защиты доступа к данным, приведена на рис. 10.11.

Таким образом, в результате этого этапа проектирования появляется достаточно подробное описание состава и функций проектируемой системы, а также информации, которую необходимо использовать в базе данных и в приложениях.

Поскольку диаграммы классов строятся на основе разработанных ранее бизнес-моделей, появляется уверенность в том, что разрабатываемая система будет действительно удовлетворять исходным требованиям заказчика.

В то же время, благодаря своему синтаксису, диаграммы классов оказываются хорошим средством структурирования и представления требований к функциональности, интерфейсам и данным для элементов проектируемой системы.

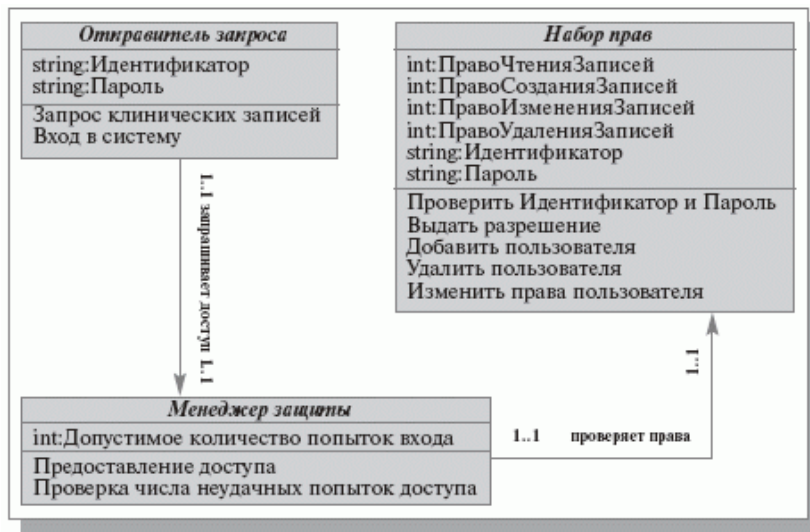


Рис. 10.11: Диаграмма классов “Защита доступа”

10.5 Разработка моделей базы данных и приложений

На этом этапе осуществляется отображение элементов полученных ранее моделей классов в элементы моделей базы данных и приложений:

- классы отображаются в таблицы;
- атрибуты – в столбцы;
- типы – в типы данных используемой СУБД;
- ассоциации – в связи между таблицами (ассоциации "многие-ко-многим" преобразуются в ассоциации "один-ко-многим" посредством создания дополнительных таблиц связей);
- приложения – в отдельные классы с окончательно определенными и связанными с данными в базе методами и атрибутами.

Поскольку модели базы данных и приложений строятся на основе единой логической модели, автоматически обеспечивается связность этих проектов (рис. 10.12).

В модель базы данных отображаются только перманентные классы, из которых удаляются атрибуты, не отображаемые в столбцах (например, атрибут типа "Общий объем продаж который получается суммированием содержимого множества полей базы данных). Некоторые атрибуты (например, АДРЕС) могут отображаться в множество столбцов (СТРАНА, ГОРОД, УЛИЦА, ДОМ, ПОЧТОВЫЙ ИНДЕКС).

Для каждого простого класса в модели базы данных формируется отдельная таблица, включающая столбцы, соответствующие атрибутам класса.

Отображение классов подтипов в таблицы осуществляется одним из стандартных способов:

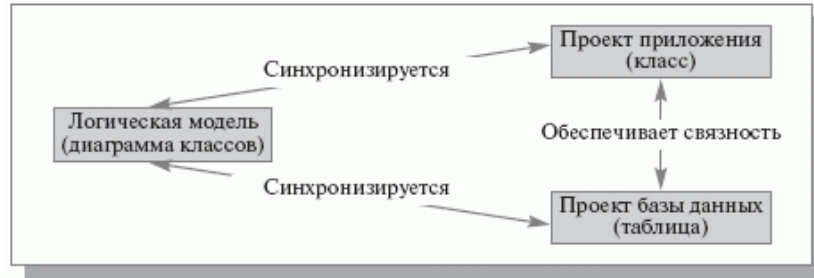


Рис. 10.12: Связь между проектами базы данных и приложений

- одна таблица на класс;
- одна таблица на суперкласс;
- одна таблица на иерархию.

В первом случае для каждого из классов создается отдельная таблица, между которыми затем устанавливаются необходимые связи. Во втором случае создается таблица для суперкласса, а затем в каждую таблицу подклассов включаются столбцы для каждого из атрибутов суперкласса. В третьем – создается единая таблица, содержащая атрибуты как суперкласса, так и всех подклассов (рис. 10.13). При этом для выделения исходных таблиц подклассов в результирующую таблицу добавляется один или более дополнительных столбцов (на рисунке показан курсивом).

Разработка проекта базы данных осуществляется с использованием специального UML-профиля (Profile for Database Design), который включает следующие основные компоненты диаграмм:

- таблица – набор записей базы данных по определенному объекту;

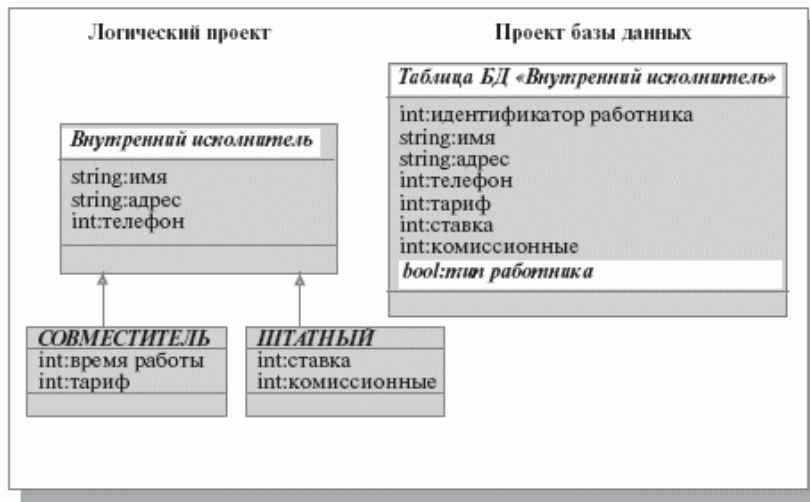


Рис. 10.13: Преобразование иерархии в таблицу

- столбец – элемент таблицы, содержащий значения одного из атрибутов таблицы;
- первичный ключ (РК) – атрибут, однозначно идентифицирующий строку таблицы;
- внешний ключ (FK) – один или группа атрибутов одной таблицы, которые могут использоваться как первичный ключ другой таблицы;
- обязательная связь – связь между двумя таблицами, при которой дочерняя таблица существует только вместе с родительской;
- необязательная связь – связь между таблицами, при которой каждая из таблиц может существовать независимо от другой;
- представление – виртуальная таблица, которая обладает всеми свойствами обычной таблицы, но не хранится постоянно в базе данных;
- хранимая процедура – функция обработки данных, выполняемая на сервере;
- домен – множество допустимых значений для столбца таблицы.

На рис. 10.14 представлен фрагмент модели базы данных — две таблицы, соответствующие классам "пациент"(рис. 10.3, рис. 10.6) и "минимальный набор данных"(рис. 10.8). Связь между ними обязательная, поскольку "минимальный набор данных" не может существовать без "пациента".

На диаграммах указываются дополнительные характеристики таблиц и столбцов:

- ограничения – определяют допустимые значения данных в столбце или операции над данными (ключ (РК,FK) – ограничение, определяющее тип ключа и его столбец; проверка (Check) – ограничение, определяющее правило контроля данных; уникальность (Unique) – ограничение, определяющее, что в столбце содержатся неповторяющиеся данные);

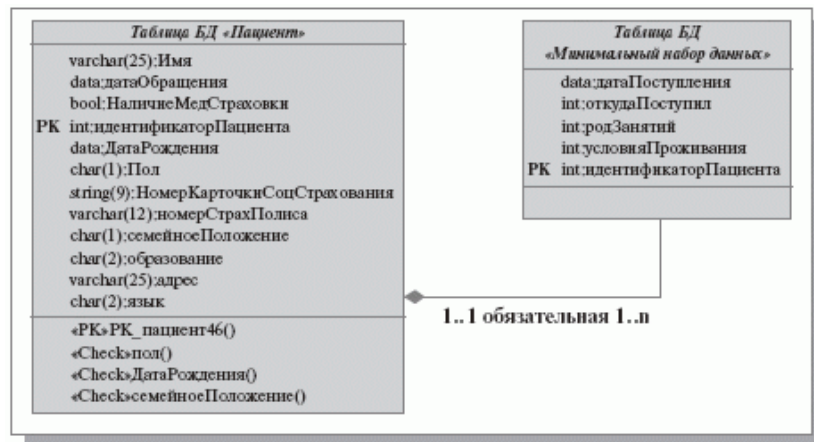


Рис. 10.14: Фрагмент модели базы данных

- триггер – программа, выполняющая при определенных условиях предписанные действия с базой данных;
- тип данных и пр.

Результатом этапа является детальное описание проекта базы данных и приложений системы.

10.6 Проектирование физической реализации системы

На этом этапе проектирования модели баз данных и приложений дополняются обозначениями их размещения на технических средствах разрабатываемой системы. На рис. 10.15 приведено изображение разделения таблицы "пациент" на три экстенда («Tablespace») в соответствии с первой буквой фамилии пациента.

Основными понятиями UML, которые используются на данном этапе, являются следующие:

- компонент – самостоятельный физический модуль системы;
- зависимость – связь между двумя элементами, при которой изменения в одном элементе вызывают изменения другого элемента;
- устройство – узел, не обрабатывающий данные;
- процессор – узел, выполняющий обработку данных;
- соединение – связь между устройствами и процессорами.

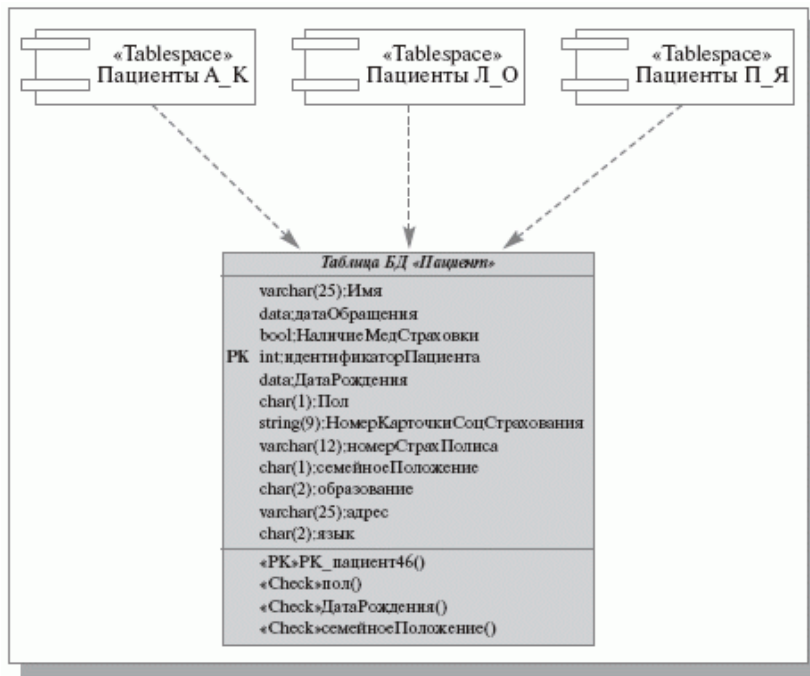


Рис. 10.15: Экстенты таблицы *Пациент*

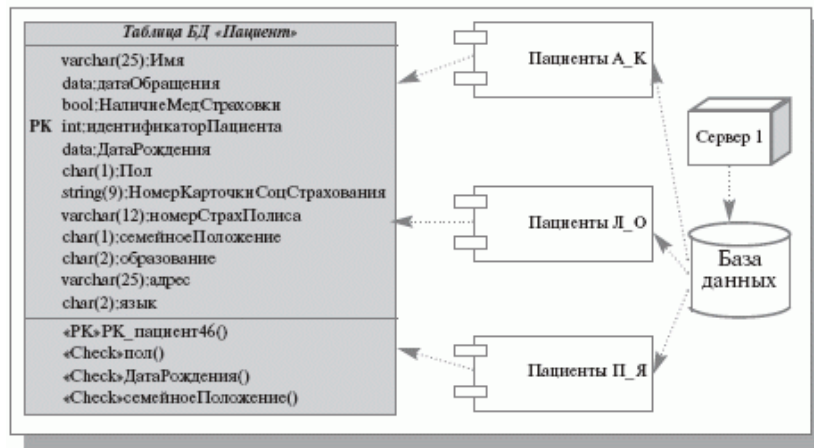


Рис. 10.16: Фрагмент диаграммы развертывания ИС

Диаграммы развертывания позволяют отобразить на единой схеме различные компоненты системы (программные и информационные) и их распределение по комплексу технических средств (рис. 10.16).

Таким образом, при проектировании сложной ИС она разделяется на части, и каждая из них затем исследуется и создается отдельно. В настоящее время используются два различных способа такого разбиения ИС на подсистемы: структурное (или функциональное) разбиение и объектная (компонентная) декомпозиция.

С позиций проектирования ИС суть функционального разбиения может быть выражена известной формулой: "Программа = Данные + Алгоритмы". При функциональной декомпозиции программной

системы ее структура описывается блок-схемами, узлы которых представляют собой "обрабатывающие центры"(функции), а связи между узлами описывают движение данных.

При объектном разбиении в системе выделяются "активные сущности"– объекты (или компоненты), которые взаимодействуют друг с другом, обмениваясь сообщениями и выполняя соответствующие функции (методы) объекта.

Если при проектировании ИС разбивается на объекты, то для ее визуального моделирования следует использовать UML. Если в основу проектирования положена функциональная декомпозиция ИС, то UML не нужен и следует использовать рассмотренные ранее структурные нотации.

В то же время, при выборе подхода к разработке ИС следует учитывать, что визуальные модели все более широко используются в существующих технологиях управления проектированием систем, сложность, масштабы и функциональность которых постоянно возрастают. Они хорошо приспособлены для решения таких часто возникающих при создании систем задач как: физическое перераспределение вычислений и данных, обеспечение параллелизма вычислений, репликация БД, обеспечение безопасности доступа к ИС, оптимизация балансировки нагрузки ИС, устойчивость к сбоям и т.п. Визуализированные средствами UML модели ИС позволяют наладить плодотворное взаимодействие между заказчиками, пользователями и командой разработчиков. Они обеспечивают ясность представления выбранных архитектурных решений и позволяют понять разрабатываемую систему во всей ее полноте.

Контрольные вопросы

1. Какие диаграммы используются на этапе описания бизнес-деятельности?

Диаграммы деятельности
Диаграммы взаимодействия
Диаграммы прецедентов
Диаграммы последовательностей
Диаграммы компонентов

2. Какие диаграммы используются на этапе описания логической модели ИС?

Диаграммы развертывания
Диаграммы видов деятельности
Диаграммы состояний
Диаграммы последовательности
Диаграммы классов
Диаграммы прецедентов

3. Какие диаграммы используются на этапе создания физической модели ИС?

Диаграммы прецедентов
Диаграммы классов
Диаграммы развертывания
Диаграммы компонентов
Диаграммы деятельности
Диаграммы последовательностей

4. Дайте определение понятию актер в UML

Разработчик проекта ИС

Личность, организация или система, взаимодействующая с ИС

Описание совокупности однородных объектов с их атрибутами, операциями, отношениями и семантикой

5. Дайте определение понятию *прецедент UML*

Законченная последовательность действий, инициированная внешним объектом (личностью или системой)

Описание совокупности однородных объектов с их атрибутами, операциями, отношениями и семантикой

Разработанный ранее прототип ИС

6. Укажите правильные свойства прецедентов

Описывает **ПОРЯДОК** выполнения действий

Описывает, **ЧТО** нужно делать

Описывает действия с точки зрения **ИСПОЛНИТЕЛЯ**

Возвращает исполнителю некоторое **СООБЩЕНИЕ**

Может описывать фрагмент действий

7. Укажите основные элементы диаграммы вида деятельности

Обозначение класса

Обозначение момента синхронизации действий

Обозначение действующего лица

Обозначение состояния

Обозначение действия

8. Укажите основные компоненты модели бизнес-объектов

Обозначение действия

Обозначение момента синхронизации действий

Обозначения внешних и внутренних исполнителей

Обозначения бизнес-сущностей, отображающие все, что используют внутренние исполнители для реализации бизнес-процессов

9. Что отражает модель системных прецедентов?

Выполнение конкретных обязанностей внутренними и внешними исполнителями с использованием ИС

Структуру базы данных ИС

Архитектуру ИС

Набрано баллов

Литература

- [1] ISO/IEC 12207:1995.
- [2] Автоматизированные Системы Стадии создания. ГОСТ 34.601-90. — 1997.
- [3] *Бек, К.* Экстремальное программирование / К. Бек. — СПб: “Питер”, 2002.
- [4] *Вендров, А. М.* Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. — М: “Финансы и статистика”, 2000.
- [5] *Гамма, Э.* Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. — СПб: “Питер”, 2007.
- [6] *Грекул, В.И.* Проектирование информационных систем / В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина. — Интернет-университет информационных технологий - ИНТУИТ.ру, 2005.
- [7] *Данилин, А.* Архитектура и стратегия. “Инь” и “янь” информационных технологий / А. Данилин, А. Слюсаренко. — Интернет-университет информационных технологий - ИНТУИТ.ру, 2005.
- [8] *Козленко, Л.* Проектирование информационных систем / Л. Козленко // *КомпьютерПресс*. — 2001. — Т. 9.

- [9] Колтунова, Е. Требования к информационной системе и модели жизненного цикла. <http://www.carabisolutions.sp.ru>.
- [10] Сузи, Р. А. Язык программирования Python / Р. А. Сузи. — Интернет-университет информационных технологий - ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2006. — С. 328.

Изучается язык программирования Python, его основные библиотеки и некоторые приложения. Рекомендовано для студентов высших учебных заведений, обучающихся по специальностям в области информационных технологий. Курс посвящен одному из бурно развивающихся и популярных в настоящее время сценарных языков программирования - Python. Язык Python позволяет быстро создавать как прототипы программных систем, так и сами программные системы, помогает в интеграции программного обеспечения для решения производственных задач. Python имеет богатую стандартную библиотеку и большое количество модулей расширения практически для всех нужд отрасли информационных технологий. Благодаря ясному синтаксису изучение языка не составляет большой проблемы. Написанные на нем программы получают структурированными по форме, и в них легко проследить логику работы. На примере языка Python рассматриваются такие важные понятия как: объектно-ориентированное программирование, функциональное программирование, событийно-управляемые программы (GUI-приложения), форматы представления данных (Unicode, XML и т.п.). Возможность диалогового режима работы интерпретатора Python позволяет существенно сократить время изучения самого языка и перейти к решению задач в соответствующих предметных областях. Python свободно доступен для многих платформ, а написанные на нем программы обычно переносимы между платформами без изменений. Это обстоятельство позволяет применять для изучения языка любую имеющуюся аппаратную платформу.

Список иллюстраций

- 2.1 Каскадная модель ЖЦ ИС
- 2.2 Поэтапная модель с промежуточным контролем
- 2.3 Спиральная модель ЖЦ ИС

- 4.1 Обобщенная схема организационного бизнес- моделирования
- 4.2 Основные этапы процессно-целевого описания компании
- 4.3 Полная бизнес-модель компании
- 4.4 Шаблон разработки миссии (матрица проекций)
- 4.5 Шаблон разработки миссии
- 4.6 Шаблон формирования бизнесов
- 4.7 Шаблон формирования бизнесов (матрица проекций)
- 4.8 Шаблон формирования основных бизнес-функций
- 4.9 Шаблон формирования основных функций менеджмента
- 4.10 Шаблон распределения функций по организационным звеньям
- 4.11 Потоксовая процессная модель
- 4.12 Функциональная схема компании
- 4.13 Схема создания Положения об организационно-функциональной структуре компании
- 4.14 Распределение функций по подразделениям производственного предприятия

4.15 Распределение функций по подразделениям торгового предприятия

5.1 Схема управления деятельностью компании

5.2 Упрощенная модель деятельности компании

5.3 Двухуровневая модель деятельности предприятия

5.4 Упрощенная матрица-генератор обеспечивающих бизнес-функций

5.5 Матрица-генератор обеспечивающих бизнес-функций

6.1 Функциональный блок

7.1 Иерархическая классификационная схема

7.2 Организационная структура подразделения предприятия-цеха отгрузки

7.3 Классификатор материальных ресурсов для обеспечения производства

7.4 Схема признаков фасетной классификации

7.5 Элементы электронного документа

8.1 Независимые от идентификации сущности

8.2 Зависимые от идентификации сущности

8.3 Графическое изображение мощности связи

8.4 Неидентифицирующая связь

8.5 Пример характеристической сущности “Хобби”

8.6 Иерархия наследования. Неполная категория

8.7 Иерархия наследования. Полная категория

8.8 Определение первичного ключа для сущности “Сотрудник”

9.1 Изображение класса в UML

9.2 Отображение связей между классами

- 9.3 Свойства ассоциации
- 9.4 Связи на диаграммах прецедентов
- 9.5 Диаграмма последовательности обработки заказа
- 9.6 Кооперативная диаграмма прохождения заказа
- 9.7 Диаграмма состояний объекта «заказ»
- 9.8 Диаграмма деятельности — обработка заказа
- 9.9 Диаграмма компонентов фрагмента КИС

- 10.1 Взаимосвязи между диаграммами UML
- 10.2 Общая диаграмма деятельности медицинского центра по обслуживанию пациента
- 10.3 Модель бизнес-прецедентов, составляющих обслуживание пациента
- 10.4 Диаграмма видов деятельности для прецедента “Оказание медицинской помощи”
- 10.5 Модель бизнес-объектов прецедента “Ответ на запрос”
- 10.6 Обобщение классов
- 10.7 Диаграмма последовательностей для прецедента “Ответ на запрос”
- 10.8 Концептуальная модель данных
- 10.9 Модель системных прецедентов
- 10.10 Диаграмма последовательностей для прецедента “Проверка прав”
- 10.11 Диаграмма классов “Защита доступа”
- 10.12 Связь между проектами базы данных и приложений
- 10.13 Преобразование иерархии в таблицу
- 10.14 Фрагмент модели базы данных
- 10.15 Экстенды таблицы *Пациент*
- 10.16 Фрагмент диаграммы развертывания ИС

Список таблиц

3.1 Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

10.1

Предметный указатель

ИС автоматические, 94
ИС автоматизированные, 94
ИС автоматизированного проектирования, 95
ИС документальные, 94
ИС фактографические, 94
ИС функциональные, 96
ИС информационно-поисковые, 95
ИС информационно-решающие, 95
ИС корпоративные, 98
ИС оперативного уровня, 96
ИС организационного управления, 94
ИС ручные, 94
ИС советующие, 96
ИС стратегическая, 96
ИС управления технологическими процессами, 95
ИС управляющие, 95

экстремальное программирование, 35
коллизия, 17
модель каскадная, 3
модель поэтапная, 3
модель спиральная, 3
рефакторинг, 39
техническое задание, 11
 типовое проектное решение, 22
жизненный цикл, 3

CRC, 39

UnitTest, 49
UserStory, 49

XP большой босс, 37
XP дипломат, 38
XP инструктор, 37

ХР консультант, 38
ХР наблюдатель, 37
ХР приёмщик, 37
ХР программист, 37
ХР разработчик, 36
ХР составитель историй, 37
ХР заказчик, 35