МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

СОВРЕМЕННЫЕ ПРОБЛЕМЫ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Электронный учебно-методический комплекс
по дисциплине в LMS Moodle

Работа выполнена по мероприятию блока 1 «Совершенствование
образовательной деятельности» Программы развития СГАУ
на 2009-2018 годы по проекту «Разработка магистерской программы «Программное обеспечение мобильных устройств»
по направлению 230100.68 «Информатика и вычислительная техника»
Соглашение №1/12 от 3.06.2013 г.

САМАРА
2013

УДК 004
С56

Автор-составитель: **Куликовских Илона Марковна**

Современные проблемы информатики и вычислительной техники [Электронный ресурс] : электрон. учеб.-метод. комплекс по дисциплине в LMS Moodle / Минобрнауки России, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); авт.-сост. И. М. Куликовских. - Электрон. текстовые и граф. дан. - Самара, 2013. – 1 эл. опт. диск (CD-ROM).

В состав учебно-методического комплекса входят:
1. Курс лекций.
2. Учебное пособие.
3. Методические указания для лабораторных работ..
4. Методические указания для практических занятий.
5. Тесты для итогового контроля знаний.
6. Рабочая программа дисциплины.

УМКД «Современные проблемы информатики и вычислительной техники» предназначен для студентов факультета информатики, обучающихся по направлению подготовки магистров 230100.68 «Информатика и вычислительная техника» в В семестре.

УМКД разработан на кафедре информационных систем и технологий.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)

Конспект лекций
по курсу
«Современные проблемы информатики и вычислительной техники»

Составитель: к.т.н., доцент кафедры ИСТ
Куликовских И.М.

Самара 2013 г.

СОДЕРЖАНИЕ

ТЕМА №1. Способы представления знаний. Data Mining. Задачи обработки текстовой информации. Классификация. Кластеризация. Метод ближайшего соседа.

Среди способов представления знаний различают словари с определениями понятий, тезаурусы, таксономии, онтологии, базы знаний.

Тезаурусом называют множество смысловыражающих единиц некоторого языка с заданной на нём системой семантических отношений. Каждому понятию сопоставляется синонимичныйдескриптор, и для дескрипторов явным образом указываются семантические отношения: род — вид, часть — целое, цель — средство и т. д.

Таксоном называют объект (понятие) некоторой предметной области. *Таксономия* (т.е. закон и упорядочение) — иерархическая структура классификаций определенного набора таксонов. В таксономиях отражены отношения "род-вид".

Онтология (в информатике) — формальное представление некоторой области знаний, включающее иерархическую структуру понятий, их связи и правила (теоремы, ограничения), принятые в этой области. Онтология вместе с набором индивидуальных экземпляров классов образует базу знаний. В действительности, трудно определить, где кончается онтология и где начинается база знаний. При этом онтологией можно называть базу однозначно понимаемых знаний.

Управление знаниями (knowledge management) — новая дисциплина, занимающаяся вопросами создания и управления знаниями, представляющими интерес для компаний. Управление знаниями определяют также как совокупность процессов, которые управляют созданием, распространением, обработкой и использованием знаний внутри предприятия. Управление знаниями включает определение ценных для компании знаний, их распространение среди сотрудников компании, использование и генерирование новых знаний.

Среди теоретических предпосылок возникновения knowledge management (КМ) можно выделить следующие.

Известно, что приобретаемый опыт в производстве изделий позволяет сокращать издержки и затраты, что связано с расширением знаний (в сфере экономики).

В сфере социологии знания генерируются, главным образом, в коллективах. На макроуровне — развиваются идеи постиндустриального, информационного или основанного на знании общества. На микроуровне исследуется поведение человека в группах и сообществах.

В философии и психологии КМ исследует различия между скрытыми и явными знаниями, между "знать как" и "знать что". Психология изучает то, как люди обучаются, забывают, действуют и т.п.

Различают корпоративные знания явные и неявные. Явные знания — это содержание документов организации таких, как письма, статьи, справочники, патенты, чертежи, программное обеспечение и т. п. Неявные (скрытые) знания — это персональные знания, связанные с индивидуальным опытом сотрудников. Часто именно скрытое знание является ключевым при принятии решении и управлении производственными процессами.

В управлении знаниями можно выделить следующие этапы:

1. Накопление, часто происходящее стихийно и бессистемно.

2. Извлечение.

3. Структурирование — выделение основных понятий, выработка способов представления информации.

4. Формализация — перевод знаний в машинный формат.

5. Сопровождение (обслуживание) — удаление, корректировка, добавление, фильтрация данных и знаний для поиска информации, необходимой пользователям.

Data Mining (DM) — направление в области интеллектуальных систем, связанное с поиском в больших объемах данных скрытых закономерностей. Data Mining можно интерпретировать как обнаружение знаний в базах данных или как интеллектуальный анализ данных. Дословно DM переводится как добыча данных. Другими словами, это добыча знаний, необходимых для принятия решений в различных сферах человеческой деятельности. При этом под знаниями понимается совокупность сведений, которая образует целостное описание, соответствующее некоторому уровню осведомленности об описываемом вопросе, предмете, проблеме и т.д. Искомые закономерности часто выражаются в виде шаблонов (паттернов — patterns), которые представляют собой некоторые выборки данных. Построение моделей прогнозирования также является целью поиска закономерностей.

Статистические методы методы анализа данных и OLAP в основном ориентированы на проверку заранее сформулированных гипотез и на предварительный анализ данных, в то время как Data Mining занимается поиском неочевидных закономерностей.

Единого мнения относительно того, какие задачи следует относить к Data Mining, нет. В большинстве источников называются следующие основные задачи:

- классификация,
- кластеризация,
- ассоциация,
- последовательность,
- прогнозирование,

Важной задачей, близкой к Data mining является поиск знаний (knowledge discovery).

С помощью классификации объекты распределяются между заранее определенными группами.

Целью кластеризации является определение таких групп.

Ассоциация имеет целью определение отношений между событиями.

Прогнозирование используется для предсказания событий на основе известных уже имевших место фактов и событий.

Text Mining — одна из подобластей Data Mining, которая ориентирована на обработку текстовой информации и широко применяется для мониторинга ресурсов Интернет. Задача Text Mining — проанализировать не синтаксис, а семантику значения текстов, выбрать из него информацию, наиболее значимую для пользователя (есть тесная связь с контент-анализом). Обычно выделяют такие приложения Text Mining:

- реферирование текстов на естественном языке;
- классификацию (тематическое индексирование) текстовых документов;
- кластеризацию текстовых документов и их фрагментов;
- построение онтологии текстового документа (основных терминов и связей между ними), например семантической сети;
- визуализация полученных знаний.

Основная особенность Data Mining — это сочетание количественного и качественного анализа. Большинство аналитических методов, используемых в технологии Data Mining, - это известные математические алгоритмы и методы.

Процесс извлечения знаний в Data Mining состоит из следующих стадий:

Стадия 1. Выявление закономерностей (свободный поиск).

Стадия 2. Использование выявленных закономерностей для предсказания неизвестных значений (прогностическое моделирование).

Стадия 3. Анализ исключений — выявление и объяснение аномалий, найденных в закономерностях.

Арсенал средств Data Mining довольно обширен. Классификация методов Data Mining выполняется по ряду признаков.

В зависимости от полноты используемых при анализе данных различают методы следующих двух групп:

1. Методы с непосредственным использованием данных с их сохранением на всех стадиях анализа. Недостаток методов этой группы — возможные сложности анализа сверхбольших баз данных. К этой группе относятся кластерный анализ, метод ближайшего соседа, метод k-ближайшего соседа, рассуждение по аналогии.

2. Методы с выявлением и использованием формализованных закономерностей, или дистилляция шаблонов. При этом образцы (шаблоны) информации извлекаются из исходных данных на стадии свободного поиска и преобразуются в некие формальные конструкции, которые и используются на стадиях прогностического моделирования и анализа исключений. Очевидно, что шаблоны значительно компактнее самих баз данных. К этой группе относятся логические методы; методы визуализации; методы кросс-табуляции; методы, основанные на уравнениях.

Статистические методы Data mining подразделяют на следующие группы:

1. Дескриптивный анализ и описание исходных данных.

2. Анализ связей (корреляционный и регрессионный анализ, факторный анализ, дисперсионный анализ).

3. Многомерный статистический анализ (компонентный анализ, дискриминантный анализ, многомерный регрессионный анализ, канонические корреляции и др.).

4. Анализ временных рядов (динамические модели и прогнозирование).

К кибернетическим методам Data Mining относят:

• искусственные нейронные сети (распознавание, кластеризация, прогноз);

• эволюционное программирование (в т.ч. алгоритмы метода группового учета аргументов);

• генетические алгоритмы (оптимизация);

• ассоциативную память (поиск аналогов, прототипов);

• нечеткую логику;

• деревья решений;

• экспертные системы.

Обработка текста в интеллектуальных системах включает морфологический, синтаксический и семантический анализ.

Морфологический анализ выполняется вне связи с контекстом, его результатами являются выделение основ слов, определение свойств слова (часть речи, падеж, число и т.п.),идентификация в множестве слов (словаре). Используют два метода морфологического анализа. Декларативный метод заключается в записи в словарь всех грамматических форм слова. Этот метод трудоемок при создании словаря, но прост при его использовании. Процедурный метод основан на записи в словарь только основ слов и выделении при собственно анализе этих основ, т.е. анализ фактически сводится к отбрасыванию аффиксов (окончаний и суффиксов) и сопоставлению оставшейся основы с содержимым словаря. Отметим, что часть слова после удаления окончания называют *токеном*.

Синтаксический анализ предназначен для определения структуры фрагментов (предложений) текста. Отметим, что в программировании синтаксическим анализом называют фазу трансляции, на которой проверяется соблюдение синтаксиса исходного языка и вырабатывается описание на некотором промежуточном языке для последующей генерации кода объектной программы

Семантический анализ — определение (в интеллектуальных системах) смысловых характеристик слов или словосочетаний. Одной из задач семантического анализа является контекстно-свободный поиск документов по запросу в виде слова или фразы в больших документальных базах. Большинство существующих систем основываются исключительно на морфологическом анализе слов и не задействуют более сложных схем анализа [3].

В управлении знаниями различают данные трех типов: сильно структурированные данные (собственно данные), слабо структурированные данные (текстовые документы на языке естественном или ограниченно естественном), информация о способах решения проблем (иногда именно эту группу называют знаниями).

Для работы с сильно структурированными данными используют технологии реляционных баз данных.

В работе с слабо структурированными данными различают несколько групп задач. К ним относятся: машинный перевод, общение человека с компьютером, синтез речи, а также задачи, рассмтриваемые ниже.

1. Поиск текстовой информации (информационный поиск). Эта задача решается в информационно-поисковых системах с использованием понятий поисковый образ и запрос и определением степени их релевантности.

Оценка релевантности чаще всего производится статистическими методами. В качестве критериев релевантности применяют:

- число совпадений слов запроса со словами в документе с учетом синонимов (предварительно выполняют морфологический анализ — выделение в словах их основы и определение грамматических характеристик слова;
- то же, но каждое совпадение имеет вес, зависящий от расстояния между ключевыми словами в документе и, возможно, от их очередности;
- то же, что и первый критерий, но вес совпадения зависит от частоты использования слова в базе документов, более редкие слова обусловливают больший вес.

Отдельное место в задаче поиска занимает *поиск по динамически формируемым запросам*. Реализация поиска по запросам, формируемым в процессе самого поиска, используется для извлечения новых фактов или формировании сообщений по вновь сформированной теме, при создании компьютерного виртуального собеседника и с необходимостью входит в число задач управления знаниями.

2. Классификация и кластеризация документов. Под *кластеризацией* понимают выделение признаков объектов некоторого множества, характеризующих степень их взаимного сходства или различия, и формирование на основе такого выделения групп (классов) родственных объектов. Собственно отнесение объектов к тому или иному классу из числа заданных называют*классификацией* .

Одним из методов выделения классообразующих признаков для текстовых документов является взвешивание терминов. Веса x_{ij} терминов в заданной выборке документов определяются одним из следующих способов:

- по наличию i-го термина (слова) в j-м документе ($x_{ij}=1$) или его отсутствия ($x_{ij}=0$);
- по частости f_{ij} появления i-го слова в j-м документе;
- по относительной частости, которую можно определить как произведение частости f_{ij} и логарифма отношения N/N_i, где N — число документов в выборке, N_i — число документов, в которых встречается i-е слово;
- по относительной частости с учетом длин документов

$$x_{ij} = \frac{f_{ij} \ln(N/N_i)}{\left[\sum_{i=1}^{m} (f_{ij} \ln(N/N_i)^2\right]^{1/2}},$$

где m — число слов в выборке.

При классификации сопоставляют входящие в систему документы с сформированными классами. Отнесение документа к определенному классу выполняется по минимуму расстояния классифицируемого документа от сформированных классов. Понятие расстояния можно связать с той или иной нормой разности векторов $\mathbf{X}_j=(x_{1j}, x_{2j},..., x_{mj})$ двух сравниваемых документов, например:

$$d_{pq} = \max_{i\in[1:m]} |x_{ip} - x_{iq}|$$

Для решения задач классификации используются алгоритмы, типичные для ИПС или систем Data Mining. Например, в Data Mining находит применение алгоритм дерева решений (Decision Tree), в соответствии с которым значение каждого из исследуемых атрибутов классифицируется с использованием правил вида “если — то”. Каждый узел дерева представляет собой некий вопрос. ответ на который позволяет отнести рассматриваемый документ к тому или иному классу.

Классы образуют путем разделения или объединения документов выборки в группы по критерию "близости" — малого расстояния между документами. Используемый при этом метод кластеризации иногда называют *методом “ближайшего соседа”*.

3. Построение тезаурусов. *Тезаурус* — упорядоченный перечень терминов, используемых в некоторой предметной области, с отражением семантических связей между ними. Существуют стандарты на требования к тезаурусам, на их структуру и правила построения (ГОСТ 7.25-80 и ГОСТ 7.24-90). Эти стандарты ориентированы на тезаурусы конкретных предметных областей, структурирование тезаурусов связано с такими понятиями, как дисциплина, предмет, метод, процесс. явление, свойство, величина, отношение и др.

4. Выражение семантики документа на формальном языке. Перевод текста с естественного языка на формальный требуется для реализации возможностей автоматической семантической обработки текста. Примерами формальных языков могут служить языки онтологий.

5. Принятие решений. Решение может быть представлено одним или совокупностью нескольких элементов заданного целевого множества. В

отличие от задачи поиска, где результатом может быть много альтернатив, здесь совокупность нескольких элементов есть одна альтернатива. Поэтому кроме отношения релевантности, нужно учитывать некоторые дополнительные отношения предпочтительности. Эти отношения задаются экспертами (как в методе анализа иерархий) или представлены функцией полезности (как в задачах оптимизации), определенной на множестве метаданных.

6. Генерация новых знаний. К новым знаниям в системах управления знаниями относится установление новых отношений на множестве элементов базы знаний (БЗ), приводящее к получению нового полезного решения возникшей практической проблемы. Это выражается в добавлении или новых продукций к базе знаний, или новых вершин и/или связей в семантическую сеть понятий. Например, установление связи документов, описывающих практические задачи, и документов, описывающих принятие решения в условиях, совпадающих с условиями задачи. Если задача принятия решений относится к интерпретации фактов при заданной базе знаний, то генерация новых знаний — изменение самой БЗ. К генерации новых знаний следует отнести извлечение информации из текстовых данных (Data Mining) и представление ее, например, в виде семантической сети.

7. **Автоматическое реферирование и автоматический машинный перевод**. Автоматический машинный перевод – это одна из старейших задач искусственного интеллекта и на текущий момент представлено множество коммерческих систем, способных переводить несложные тексты.

Пусть в обучающей выборке имеется множество $\mathbf{Q}$ объектов, объект $q_j \in \mathbf{Q}$ характеризуется вектором параметров $\mathbf{X}_j=(x_1,x_2,x_3,...x_n)$, объекты распределены между кластерами $\mathbf{C}_i$ множества $\mathbf{C}$.

Примечание 1

Далее объектами будем считать векторы $\mathbf{X}_j$.

При задании нового объекта в виде вектора $\mathbf{X}_r$ нужно отнести его к одному из кластеров множества $\mathbf{C}$. Другими словами, *классификация* заключается в определении $\mathbf{C}_i$ по заданному $\mathbf{X}_r$.

Одним из методом решения задачи классификации является метод, основанный на формуле вероятностей гипотез (формуле Байеса), в соответствии с которой вероятность выбора гипотезы $\mathbf{C}_i$ для заданного $\mathbf{X}_r$ вычисляется как

$$p(\mathbf{C}_i|\mathbf{X}_r) = p(\mathbf{X}_r|\mathbf{C}_i)\, p(\mathbf{C}_i) / P(\mathbf{X}_r),$$

где $p(\mathbf{X}_r|\mathbf{C}_i)$ — вероятность появления объекта $\mathbf{X}_r$ в кластере $\mathbf{C}_i$, $p(\mathbf{C}_i)$ — вероятность того, что произвольный объект обучающей выборки отнесен к кластеру $\mathbf{C}_i$; $P(\mathbf{X}_r) = \sum_{i=1}^{m} p(\mathbf{X}_r|\mathbf{C}_i)p(\mathbf{C}_i)$ — вероятность того, что объект обучающей выборки есть $\mathbf{X}_r$.

Если $\mathbf{X}_j$ числовые векторы, то классификация может выполняться на основе сопоставления расстояний от объекта до центров кластеров с отнесением объекта к наиболее близкому кластеру.

Если при классификации документов кластеры являются тематическими, т.е. документ требуется отнести к одной из рубрик (тем), то классификация возможна по степени близости тезаурусных проекций кластеров и классифицируемого документа. В этом случае предварительно разрабатываются предметные онтологии (или тезаурусы) для каждой рубрики (темы). Тезаурусная проекцияопределяется вектором $\mathbf{Y}_k = (y_{1k},...y_{nk})$, где y_{ik}- характеризует наличие или частотность i -го дескриптора A_i в документах k-го кластера, n - суммарное число разных дескрипторов в используемых онтологиях. Степень r_{jk} соответствия j-го документа $\mathbf{X}_j=(x_1,x_2,x_3,...x_n)$ k-му кластеру можно оценить, например, по косинусу угла между векторами $\mathbf{X}_j$ и $\mathbf{Y}_k$:

$r_{jk} = \mathbf{X}_j * \mathbf{Y}_k/(|\mathbf{X}_j|*|\mathbf{Y}_k|)$.

К средствам классификации относят также деревья решений и правила. Деревья состоят из вершин ИЛИ, соответствующих параметрам x_k, исходящие из вершины ИЛИ дуги соответствуют альтернативам — возможным значениям параметра x_k, в вершинах для выбора альтернативы используются правила ЕСЛИ…ТО... Правила формируются по имеющейся обучающей выборке. Терминальные вершины соответствуют гипотезам $\mathbf{C}_i$.

Кластеризация — разделение множества $\mathbf{Q}$ объектов на группы (классы, кластеры) по тем или иным признакам сходства объектов, т.е. формирование множества $\mathbf{C}$ классов для объектов некоторой предметной области: $\mathbf{Q} \rightarrow \mathbf{C}$. Нужно выбрать множество $\mathbf{X}$ признаков (параметров) объектов, по которым определяется сходство объектов, и обучающую выборку.

Если множество $\mathbf{X}$ метризовано, то оценкой близости двух объектов r и s будет норма вектора $|\mathbf{X}_r - \mathbf{X}_s|$ (расстояние между r и s). Например:

$d_{rs} = \max |\mathbf{X}_{ir} - \mathbf{X}_{is}|$ при $i \in [1:m]$.

Межкластерное расстояние – расстояние между центрами кластеров. Центр k-го кластера имеет усредненные координаты

$$\mathbf{X}_{kц} = \sum_{i=1}^{n} \mathbf{X}_{ik} / n,$$

где $\mathbf{X}_{ik}$ – вектор параметров i-го объекта, входящего в. k-й кластер.

Один из алгоритмов кластеризации:

1) Все объекты обучающей выборки помещаются в первый кластер;

2) Объект с наибольшим усредненным расстоянием от других объектов переносится во второй кластер;

3) Все объекты с меньшим средним расстоянием до объектов второго кластера, чем до первого, переносятся во второй кластер;

4) Если наибольший диаметр кластеров больше заданного порога, то для кластера с большим диаметром повторяются пункты 1-3, иначе останов. Диаметр кластера определяется как наибольшщее расстояние между парой объектов, принадлежащих кластеру:

$$\text{Диаметр} = \max_{r,s} d_{rs}$$

Если множество $\mathbf{X}$ неметризовано, то расстоянием между двумя объектами r и s будет число несовпадающих элементов в множествах $\mathbf{X}_r$ и $\mathbf{X}_s$.

Метод ближайшего соседа (nearest neighbour) относится к методам классификации. Классифицируемый объект сравнивается с уже имеющимися прецедентами и принимается решение об отнесении объекта к тому кластеру, к которому принадлежит ближайший прецедент. Напомним, что *прецедентом* в интеллектуальных системах называют описание ситуации в сочетании с подробным указанием действий, предпринимаемых в данной ситуации. Понятие близости для каждого приложения нужно предварительно конкретизировать.

Метод k-ближайшего соседа — разновидность метода ближайшего соседа, в нем в качестве ближайших соседей рассматриваются не один, а k ближайших соседей.

Обычно в памяти хранятся не все прецеденты, а только подмножество "типичных" случаев. Тогда метод называют *методом рассуждения по аналогии* (Case Based Reasoning, CBR), или рассуждения на основе аналогичных случаев, или рассуждения по прецедентам.

Подход, основанный на прецедентах, условно можно разделить на следующие этапы:

• сбор подробной информации о поставленной задаче;

• сопоставление этой информации с деталями прецедентов, хранящихся в базе, для выявления аналогичных случаев;

• выбор прецедента, наиболее близкого к текущей проблеме, из базы прецедентов;

• адаптация выбранного решения к текущей проблеме, если это необходимо;

• проверка корректности каждого вновь полученного решения;

• занесение детальной информации о новом прецеденте в базу прецедентов.

Таким образом, вывод, основанный на прецедентах, представляет собой такой метод анализа данных, который делает заключения относительно данной ситуации по результатам поиска аналогий, хранящихся в базе прецедентов.

Метод ближайшего соседа довольно прост для применения. Недостатки метода заключаются в сложности выбора меры близости объектов, а также в высокой трудоемкости использования, поскольку при распознавании требуется полный перебор прецедентов.

ТЕМА №2. Метод анализа иерархий. Онтологии. Средства построения онтологий. Средства построения онтологий. Системы управления знаниями. Онтологическая СУЗ.

К числу методов принятия решений, используемых в СППР и СУЗ, относится метод *анализа иерархий*. Он предназначен для выбора одной из множества возможных альтернатив в условиях неопределенности, т.е. в условиях использования неточных, неполных, неколичественных исходных данных. Метод анализа иерархий основан на предпочтениях экспертов.

Применение метода сводится к выполнению следующих процедур

1. Построение иерархии целей и показателей в виде дерева (сети), в котором исходная цель есть корень, на последующих иерархических уровнях (ярусах) находятся подцели, критерии, а на нижнем ярусе располагаются альтернативы искомого решения.

2. Выявление предпочтений вершин (вершины имеют характер ИЛИ-вершин) на последующих ярусах. Для этого для каждой группы альтернативных вершин составляются матрицы парных сравнений. Элементом C_{ij} матрицы парных сравнений альтернативных вершин на некотором иерархическом уровне является балльная оценка эксперта (или усредненная оценка нескольких экспертов) предпочтительности вершины i по сравнению с вершиной j. При этом $C_{ij} = 1/C_{ji}$. Результирующей оценкой предпочтительности альтернативных вершин по отношению друг к другу является главный собственный вектор матрицы парных сравнений $\mathbf{Q}$, определяемый в результате решения уравнения

$$\mathbf{CQ}=\lambda_{max}\mathbf{Q},$$

где $\mathbf{C}$ матрица парных сравнений, λ_{max} — максимальное собственное значение матрицы $\mathbf{C}$.

Примечание 1

Отметим, что приближенно элементы главного собственного вектора определяются суммированием элементов строк матрицы.

3. На каждом ярусе может быть несколько групп альтернативных вершин и столько же векторов $\mathbf{Q}$. Для примера рис. 1 на первом ярусе имеется одна группа вершин, обозначенная 01, на втором ярусе — две группы 11 и 12, на третьем ярусе — три группы 21, 22 и 23. Каждой группе соответствует

матрица парных сравнений со своим собственным вектором ($\mathbf{Q}_{21}$ для матрицы вершин группы 21, $\mathbf{Q}_{22}$ — для матрицы вершин группы 22 и т.д.). Пересчет собственных векторов $\mathbf{Q}_{ij}$ в вектора предпочтительности $\mathbf{G}_m$ вершин по отношению к вершинам нижнего яруса выполняется по формуле

$$\mathbf{G}_m = \mathbf{A}\mathbf{Q}_{ij},$$

где $\mathbf{A}$ — матрица, столбцами которой являются вектора $\mathbf{G}$ смежных вершин соседнего нижнего яруса.

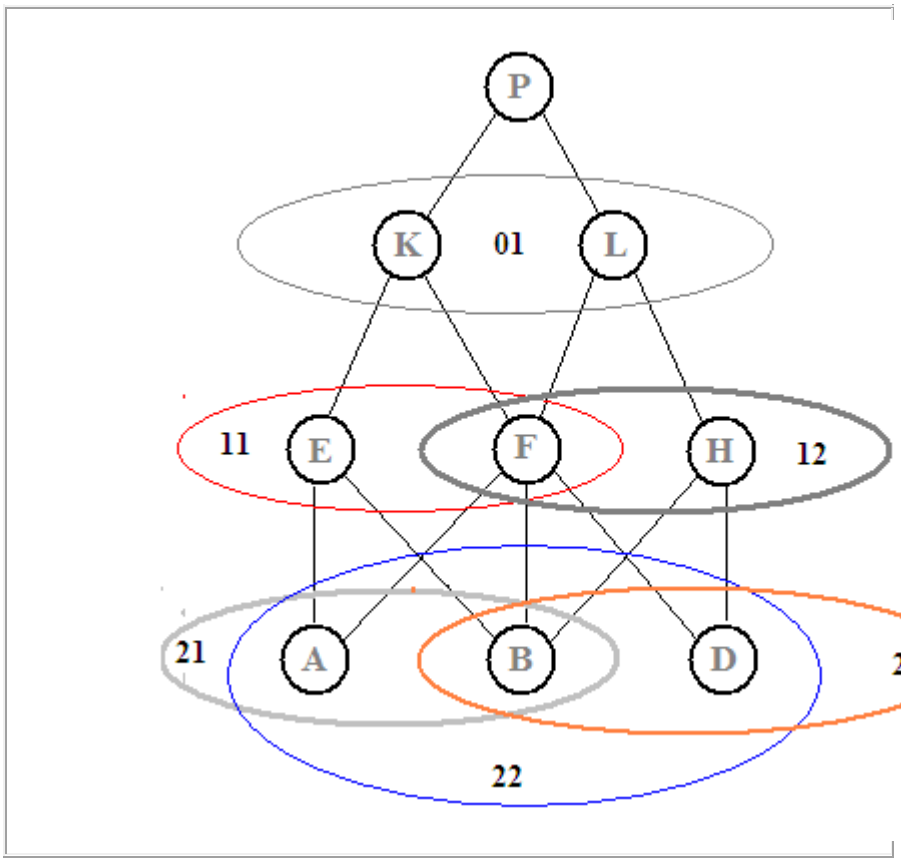


Рис. 1. Пример иерархического дерева

Для примера рис.1 имеем

$$\mathbf{G}_E = \mathbf{Q}_{21};\ \mathbf{G}_F = \mathbf{Q}_{22};\ \mathbf{G}_H = \mathbf{Q}_{23};$$

$$\mathbf{G}_K = [\mathbf{G}_E, \mathbf{G}_F]\,\mathbf{Q}_{11};\ \mathbf{G}_L = [\mathbf{G}_F, \mathbf{G}_H]\,\mathbf{Q}_{12};$$

$$\mathbf{G}_P = [\mathbf{G}_K, \mathbf{G}_L]\,\mathbf{Q}_{01}.$$

Трехэлементный вектор $\mathbf{G}_P$ в примере выражает предпочтения альтернатив, представленных вершинами A, B, D.

Существует несколько определений *онтологии*. Дословный перевод от древнегреческого (греч. on, ontos — сущее, logos — учение) — наука о сущем. Термин "Онтология" был предложен Р. Гоклениусом в 1613 г. и обозначал раздел философии, изучающий бытие.

В искусственном интеллекте и информатике онтология – это формальное описание понятий (классов) в рассматриваемой предметной области, свойств каждого понятия (атрибутов, слотов, ролей), включает также декларативные и процедурные интерпретации понятий и их отношений и ограничения (фасеты), наложенные на слоты. Другими словами, онтология – это конкретный способ выражения значений, подразумеваемых у совместно используемой лексики. Онтология рассматривается и как раздел прикладной лингвистики, и как раздел искусственного интеллекта.

В центре большинства онтологий находятся классы. Слоты могут иметь различные фасеты, которые описывают тип значения, разрешенные значения, число значений (мощность) и др.

Другое определение онтологии $\mathbf{S}$ дается следующей ее моделью:

$\mathbf{S} = (\mathbf{X}, \mathbf{R}, \mathbf{F})$, (1)

где $\mathbf{X}$ — множество понятий предметной области, называемых также *концептами*, $\mathbf{R}$ — множество отношений между концептами, $\mathbf{F}$ — множество функций интерпретации концептов и отношений.

Частные случаи (1):

- Простой словарь $\mathbf{R} = \varnothing$, $\mathbf{F} = \varnothing$; словари часто называют глоссариями, в них наряду с самими концептами описываются грамматические, стилистические характеристики и примеры использования.
- Простая таксономия (т.е. иерархическая система понятий) $\mathbf{F} = \varnothing$.

Часто считают, что в состав онтологий входят также аксиомы, т.е. некоторые утверждения, которые невозможно выразить лишь через понятия и отношения. Кроме того, из классов (понятий) выделяют экземпляры классов, т.е. некоторые объекты, существующие в единственном экземпляре.

Важно различать класс и его имя: классы представляют понятия предметной области, а не слова, которые обозначают эти понятия. Синонимы одного и того же понятия не представляют различные классы.

Онтологии формально схожи с XML Schema, но отличаются тем, что онтологии являются представлением знаний, а не форматом сообщений.

Для представления онтологий применяют дескриптивную логику, логику первого порядка, графы и семантические сети.

Язык описания онтологий — формальный язык, используемый для кодирования онтологии. Наиболее известные среди них: OWL — ontology web language, стандарт W3C, язык для семантических утверждений, разработанный как расширение RDF и RDFS; KIF (Knowledge Interchange Format или формат обмена знаниями) — основанный на S-выражениях синтаксис для логики; CycL — онтологический язык, использующийся в проекте Cyc, основан

на исчислении предикатов с некоторыми расширениями более высокого порядка; DAML+OIL.

Обычно разработка онтологии включает:

• определение понятий;

• расположение понятий в таксономическом порядке (подкласс – надкласс);

• определение слотов и описание допускаемых значений этих слотов;

• заполнение значений слотов экземпляров.

Не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. Лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений. Разработка онтологии – это итерационный процесс, причем обычно этот процесс должен продолжаться в течение всего жизненного цикла онтологии.

Различают варианты разработки онтологий:

- нисходящий — разработка начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий;
- восходящий — разработка начинается с определения самых конкретных классов, листьев иерархии, с последующей группировкой этих классов в более общие понятия;
- комбинированный — это сочетание нисходящего и восходящего подходов, сначала определяются наиболее заметные понятия, которые затем соответствующим образом обобщаются и ограничиваются.

Основные применения онтологий: семантический поиск информации (включая поиск ответов на вопросы), создание баз знаний, автоматическая рубрикация документов, реализация процедур вывода и др.

Для создания и поддержки онтологий разработан ряд программных продуктов, которые выполняют редактирование, просмотр, документирование онтологий, импорт и экспорт онтологий между системами и другие функции управления онтологиями.

Основная функция любого редактора онтологий состоит в поддержке процесса формализации знаний и представлении онтологии как спецификации (точного и полного описания).

Система *Ontolingua* была разработана в KSL (Knowledge Systems Laboratory) Стенфордского университета и стала первым инструментом инженерии онтологий. Она состоит из сервера и языка представления знаний.

Сервер Ontolingua организован в виде набора онтологий, относящихся к Web-приложениям, которые надстраиваются над системой

представления знаний Ontolingua. Редактор онтологий – наиболее важное приложение сервера Ontolingua является Web-приложением на основе форм HTML. Кроме редактора онтологий, сервер Ontolingua включает сетевое приложение Webster (получение определений концептов), сервер OKBC (доступ к онтологиям Ontolingua по протоколу OKBC) и Chimaera (анализ, объединение, интегрирование онтологий). Все приложения, кроме сервера OKBC, реализованы на основе форм HTML. Система представления знаний реализована на Lisp.

Сервер Ontolingua также предоставляет архив онтологий, включающий большое количество онтологий различных предметных областей, что позволяет создавать онтологии из уже существующих. Сервер поддерживает совместную разработку онтологии несколькими пользователями, для чего используются понятия пользователей и групп. Система включает графический браузер, позволяющий просмотреть иерархию концептов, включая экземпляры. Ontolingua обеспечивает использование принципа множественного наследования и богатый набор примитивов. Сохраненные на сервере онтологии могут быть преобразованы в различные форматы для использования другими приложениями, а также импортированы из ряда языков в язык Ontolingua.

Protégé – локальная, свободно распространяемая Java-программа, разработанная группой медицинской информатики Стенфордского университета (первая версия – 1987, последняя Protégé-2.1.1 – июнь 2004). Программа предназначена для построения (создания, редактирования и просмотра) онтологий прикладной области. Её первоначальная цель – помочь разработчикам программного обеспечения в создании и поддержке явных моделей предметной области и включение этих моделей непосредственно в программный код. Protégé включает редактор онтологий, позволяющий проектировать онтологии, разворачивая иерархическую структуру абстрактных или конкретных классов и слотов. Структура онтологии сделана аналогично иерархической структуре каталога. На основе сформированной онтологии, Protégé может генерировать формы получения знаний для введения экземпляров классов и подклассов. Инструмент имеет графический интерфейс, удобный для использования неопытными пользователями, снабжен справками и примерами.

Protégé основан на фреймовой модели представления знания OKBC (Open Knowledge Base Connectivity) и снабжен рядом плагинов, что позволяет его адаптировать для редактирования моделей, хранимых в разных форматах (стандартный текстовый, в базе данных JDBC, UML, языков XML, XOL, SHOE, RDF и RDFS, DAML+OIL, OWL).

OntoEdit первоначально был разработан в институте AIFB (Institute of Applied Informatics and Formal Description Methods) Университета Karlsruhe (сейчас коммерциализован Ontoprise GmbH) выполняет проверку, просмотр, кодирование и модификацию онтологий. В настоящее время OntoEdit поддерживает языки представления: FLogic, включая машину вывода, OIL, расширение RDFS и внутреннюю, основанную на XML, сериализацию модели онтологии, используя OXML — язык представления знаний OntoEdit (OntoEdit's XML-based Ontology representation Language). К достоинствам инструмента можно отнести удобство использования; разработку онтологии под руководством методологии и с помощью процесса логического вывода; разработку аксиом; расширяемую структуру посредством плагинов, а также очень хорошую документацию.

Так же как и Protégé, OntoEdit – автономное Java–приложение, которое можно локально установить на компьютере. Свободно распространяемая версия OntoEdit Free ограничена 50 концептами, 50 отношениями и 50 экземплярами. Архитектура OntoEdit подобна Protégé.

OilEd – автономный графический редактор онтологий, разработан в Манчестерском университете в рамках европейского IST проекта On-To-Knowledge. Инструмент основан на языке OIL (сейчас адаптирован для DAML+OIL, в перспективе – OWL), который сочетает в себе фреймовую структуру и выразительность дескриптивной логики (DL -Description Logics) с сервисами рассуждения. Что позволило обеспечить понятный и интуитивный стиль интерфейса пользователя и преимущества поддержки рассуждения (обнаружение логически противоречивых классов и скрытых отношений подкласса).

Из недостатков можно выделить отсутствие поддержки экземпляров. Существующая версия не обеспечивает полную среду разработки – не поддерживается разработка онтологий большого масштаба, миграция и интеграция онтологий, контроль версий и т.д. OilEd можно рассматривать как "NotePad" редакторов онтологий, предлагающий достаточную функциональность, чтобы позволить пользователям строить онтологии и продемонстрировать, как можно использовать механизм рассуждения FaCT для проверки онтологии на непротиворечивость.

В последнее время наблюдается рост популярности редактора OilEd. Он используется как для обучения, так и для исследования. Инструмент свободно распространяется по общедоступной лицензии GPL.

WebOnto разработан для Tadzebao – инструмента исследования онтологий и предназначен для поддержки совместного просмотра, создания и

редактирования онтологий. Его цели – простота использования, предоставление средств масштабирования для построения больших онтологий.

Для моделирования онтологий WebOnto использует язык OCML (Operational Conceptual Modeling Language). В WebOnto пользователь может создавать структуры, включая классы с множественным наследованием, что можно выполнять графически. Все слоты наследуются корректно. Инструмент проверяет вновь вводимые данные контролем целостности кода OCML.

Инструмент имеет ряд полезных особенностей: сохранение структурных диаграмм, раздельный просмотр отношений, классов, правил и т.д. Другие возможности включают совместную работу нескольких пользователей над онтологией, использование диаграмм, функций передачи и приёма и др.

OntoSaurus является Web-браузером для баз знаний LOOM. Он состоит из двух основных модулей: сервера онтологий и Web-браузера для редактирования и просмотра онтологий LOOM с помощью HTML-форм, обеспечивая для них графический интерфейс. OntoSaurus также предоставляет ограниченные средства редактирования, но его основная функция — просмотр онтологий. Но для построения сложных онтологий нужно понимать язык LOOM. Большинство пользователей строят онтологию на языке LOOM в другом редакторе, а затем для просмотра и редактирования импортируют его в OntoSaurus. В OntoSaurus реализованы все возможности языка LOOM. Обеспечиваются автоматический контроль совместимости, дедуктивная поддержка рассуждения и некоторые другие функции.

Конструктор онтологий ODE (Ontological Design Environment), который взаимодействует с пользователями на концептуальном уровне в отличие от инструментов, подобно OntoSaurus, общающихся на символьном уровне. Мотивом для ODE послужило то, что людям проще формулировать онтологии на концептуальном уровне. ODE обеспечивает пользователей набором таблиц для заполнения (концептов, атрибутов, отношений) и автоматически генерирует для них код в LOOM, Ontolingua и FLogic.

KADS22 — инструмент поддержки проектирования моделей знаний согласно методологии CommonKADS. Онтологии составляют часть таких моделей знаний (другая часть — модели вывода). Модели CommonKADS определены в CML (Conceptual Modeling Language). KADS22 – интерактивный графический интерфейс для CML со следующими функциональными возможностями: синтаксический анализ файлов CML, печать, просмотр гипертекста, поиск, генерация глоссария и генерация HTML.

Инструменты для отображения, выравнивания и объединения онтологий

Сегодня онтологии доступны в разных представлениях. Но, что делать, когда мы находим несколько онтологий, которые бы хотели использовать, но они не соответствуют друг другу?

Тогда используют средства отображения, выравнивания и объединения онтологий, которые нужны для того, чтобы не только вводить и редактировать онтологическую информацию, но и анализировать ее, выполняя типичные операции над онтологиями, например:

- объединение (merging) онтологий — операция, которая по двум онтологиям генерирует третью, объединяющую информацию из первых двух;
- отображение (mapping) одной онтологии на другую — нахождение семантических связей между подобными элементами разных онтологий;
- выравнивание (alignment) онтологий — установка различного вида соответствий между двумя онтологиями для того, чтобы они могли использовать информацию друг друга.

Инструменты объединения онтологий помогают пользователям найти сходство и различие между исходными онтологиями и создают результирующую онтологию, которая содержит элементы исходных онтологий. Для достижения этой цели они автоматически определяют соответствия между концептами в исходных онтологиях или обеспечивают среду, где пользователь может легко найти и определить эти соответствия. Эти инструменты известны как инструменты отображения, выравнивания и объединения онтологий, так как они выполняют сходные операции для процессов отображения, выравнивания и объединения.

Отображение (mapping) онтологии заключается в нахождении семантических связей подобных элементов из разных онтологий.

Выравнивание (alignment) онтологий состоит в том, чтобы установить различные виды соответствия (или связи) между двумя онтологиями, а затем повторно сохранить исходные онтологии и таким образом, в дальнейшем использовать информацию друг друга. Объединение (merging) онтологий – генерация одной согласованной онтологии из двух исходных.

Исследователи разных областей информатики работают над автоматическим или поддерживаемым инструментально объединением онтологий (или иерархии классов, или объектно-ориентированных схем, или схем баз данных – определенная терминология изменяется в зависимости от области применения). Однако и автоматическое объединение онтологий, и создание инструментальных средств, которые бы управляли пользователем в этом процессе, находятся на ранних стадиях развития. В этом разделе представлен краткий обзор некоторых из существующих подходов.

Инструментальные средства, которые имеют дело с нахождением соответствия между онтологиями, классифицируются следующим образом:

для объединения двух онтологий с целью создания одной новой (PROMPT, Chimaera, OntoMerge);

для определения функции преобразования из одной онтологии в другую (OntoMorph);

для определения отображения между концептами в двух онтологиях, находя пары соответствующих концептов (например, OBSERVER, FCA-Merge);

для определения правил отображения для связи только релевантных частей исходных онтологий (ONION).

Рассмотрим теперь вышеупомянутые средства более подробно.

PROMPT — дополнение к системе Protégé, реализованное в виде плагина, служит для объединения и группировки онтологий. При объединении двух онтологий PROMPT создает список предлагаемых операций. Операция может состоять, например, из объединения двух терминов или копирования терминов в новую онтологию. Пользователь может выполнить операцию, выбирая одну из предлагаемых или определяя непосредственно операцию. PROMPT выполняет выбранную операцию и дополнительные изменения, вызванные этой операцией. Потом список предлагаемых операций модифицируется и создается список конфликтов и возможных решений этих конфликтов. Это повторяется до тех пор, пока не будет готова новая онтология.

Chimaera — интерактивный инструмент для объединения, основанный на редакторе онтологий Ontolingua. Chimaera позволяет пользователю объединять онтологии, разработанные в различных формализмах. Пользователь может запрашивать анализ или руководство от Chimaera в любой момент в течение процесса объединения, и инструмент направит его на те места в онтологии, где требуется его вмешательство. В своих предложениях Chimaera главным образом полагается на то, из какой онтологии прибыли концепты, основываясь на их именах. Chimaera оставляет решение о том, что делать пользователю, и не делает никаких предложений самостоятельно. Единственное таксономическое отношение, которое рассматривает Chimaera – отношение подкласс — суперкласс. Chimaera самый близкий к PROMPT. Однако поскольку он использует в своем анализе только иерархию класса, он пропускает многие из соответствий, которые находит PROMPT. Эти соответствия включают предложения по объединению слотов с подобными именами, которые относятся к объединенным классам, объединению доменов слотов, которые были объединены и т. д.

В OntoMerge объединенная онтология есть объединение двух исходных онтологий и набора аксиом соединения. Первый шаг в процессе объединения в OntoMerge состоит в трансляции обеих онтологий к общему синтаксическому представлению на разработанном авторами языке. Затем инженер онтологии определяет аксиомы соединения, содержащие термины из обеих онтологий. Процесс трансляции экземпляров выглядит следующим образом: все экземпляры в исходных онтологиях, рассматриваются как находящиеся в объединенной онтологии. Затем на основе инструкций в исходных онтологиях и аксиом соединения машина вывода сделает заключение, таким образом, создавая новые данные в объединенной онтологии. OntoMerge предоставляет инструменты для трансляции данных-экземпляров в объединенную онтологию.

OntoMorph определяет набор операторов преобразования, которые можно применить к онтологии. Затем человек-эксперт использует начальный список пар и исходных онтологий для определения набора операторов, которые должны быть применены к исходным онтологиям для устранения различий между ними, и OntoMorph применяет эти операторы. Таким образом, совокупность операций может выполняться за один шаг. Однако, человек-эксперт не получает никакого руководства за исключением начального списка пар.

Система OBSERVER применяет дескриптивную логику (DL) для ответа на запросы, используя несколько онтологий и информацию об отображении между ними. Вначале пользователи определяют набор межонтологических отношений. Система помогает справиться с этой задачей, находя синонимы в исходных онтологиях. Определив отображения, пользователи могут формулировать запросы в терминах DL с помощью собственной онтологии. Затем OBSERVER использует информацию отображения для формулировки запросов к исходным онтологиям. OBSERVER в значительной степени полагается на тот факт, что описания в онтологиях и запросах являются содержательными.

FCA-Merge — метод для сравнения онтологий, которые имеют набор общих экземпляров или набор общих документов, аннотируемых с помощью концептов исходных онтологий. Основываясь на этой информации, FCA-Merge использует математические методы из Formal Concept Analysis для того чтобы произвести решетку концептов, связывающую концепты исходных онтологий. Алгоритм предлагает отношения эквивалентности и подкласс-суперкласс. Затем инженер онтологии может анализировать результат и использовать его как руководство для создания объединенной онтологии. Однако предположение, что две объединяемые онтологии используют общий набор

экземпляров или имеют набор документов, в котором каждый документ аннотируется терминами обоих источников слишком жесткое и на практике такая ситуация происходит редко. В качестве альтернативы, авторы предлагают использовать методы обработки естественного языка для аннотации набора документов концептами из этих двух онтологий.

Система ONION (ONtology compositION) основана на алгебре онтологии. Поэтому, она предоставляет инструменты для определения правил артикуляции (соединения) между онтологиями. Правила артикуляции обычно учитывают только релевантные части исходных онтологий. Для того чтобы предложить соединение, ONION использует и лексические методы, и методы на основе графов. Метод нахождения лексического подобия между именами концептов использует словари и методы семантической индексации, основанные на местонахождении группы слов в тексте.

Инструменты аннотирования на основе онтологий

Важнейшим предусловием реализации целей семантического Web является возможность аннотировать Web-ресурсы семантической информацией. В связи с этим в последние годы инструменты инженерии онтологий эволюционируют в сторону разработки инструментов аннотирования на основе онтологий.

Инструмент аннотации MnM обеспечивает поддержку автоматической и полуавтоматической разметки Web-страниц семантическим содержанием. MnM интегрирует Web-браузер и редактор онтологии и обеспечивает открытые интерфейсы связи с серверами онтологий и инструментами извлечения информации. MnM можно рассматривать в качестве одного из первых примеров следующего поколения редакторов онтологий, на основе Web, ориентирующихся на семантическую разметку и обеспечивающих механизм полномасштабной автоматической разметки Web-страниц.

С помощью SHOE’s Knowledge Annotator пользователь может также описывать содержание Web-страниц. Инструмент имеет интерфейс, который отображает экземпляры, онтологии и утверждения (собранные документы). Также обеспечивается проверка целостности. SHOE’s Knowledge Annotator позволяет пользователям выполнять разметку страниц в SHOE, под управлением онтологий доступных локально или через URL. Эти размеченные страницы могут быть проанализированы инструментальными средствами, знающими язык SHOE, типа SHOE Search. Аннотируемые Web-страницы могут быть также проанализированы другим инструментом по имени Expos'e, а содержание будет сохранено в репозитарии. Это SHOE-знание затем сохраняется в базе знаний Parka.

Инструмент Metabrowser также частично решает проблему аннотирования Web-ресурсов. Он может работать, например, на базе онтологии Дублинского ядра (Дублинское ядро можно рассматривать как простейшую онтологию) и предлагать ряд возможностей для автоматического создания и просмотра метаданных. Metabrowser (включая свободно распространяемую версию), отображает метаданные Web-страницы вместе с самой Web-страницей.

Известны и российские разработки, например, системы CAKE, ВИКОНТ, VITA, позволяющих визуально проектировать онтологии различных предметных областей, или система БиГОР для сопровождения онтологий, как составных частей образовательных ресурсов.

Сравнительный анализ инструментов

Мы вкратце рассмотрели три группы инструментов: построения онтологий; отображения, выравнивания и объединения онтологий и аннотирования на основе онтологий. В соответствии с каждой группой инструментов попытаемся сравнить их между собой

Инструменты построения онтологий можно разделить на два типа: разработанные для редактирования онтологий на определенном языке онтологий и интегрированные наращиваемые инструментальные сайты (Web-приложения, на основе форм HTML и/или Java-апплетов), большинство из которых не зависит от языка представления.

Следует подчеркнуть, что большинство из рассмотренных инструментальных средств разрабатываются университетскими исследовательскими группами, которые предоставляют открытый код, либо предлагают свободный доступ к функциям. Однако наиболее перспективные из них передаются коммерческим компаниям (например, OntoEdit Professional — лицензированный продукт).

Инструменты OntoEdit, WebODE и KADS22 дают поддержку методологиям построения онтологий, соответственно On-To-Knowledge, METHONTOLOGY и CommonKADS, что не мешает им быть используемыми в других методологиях или вообще без них.

Касаясь технического аспекта, а именно архитектуры программного обеспечения (локальная, клиент-серверная, n-уровневая), расширяемости, языков программирования, на которых реализованы инструменты, способов хранения онтологий (в файлах или базах данных), необходимо отметить следующее.

Более ранние инструменты Ontolingua, OntoSaurus и WebOnto имеют клиент-серверную архитектуру. Protégé, OntoEdit и OilEd имеют 3-х уровневую

архитектуру, где существует четкое разделение между хранением онтологий, модулями бизнес-логики, логики приложений и приложениями интерфейса пользователя. Эти инструменты обладают большими возможностями по наращиванию (например, при помощи плагинов). Большинство инструментов хранит свои онтологии в текстовых файлах, что ограничивает размер онтологий. Только Protégé и WebODE могут хранить свои онтологии в базах данных и таким образом управлять большими онтологиями. Наконец, большинство инструментов реализовано на Java.

Выше уже говорилось о том, что модели знания инструментов определяют компоненты, которые должны использоваться при построении онтологии. Большинство инструментов представляет онтологии, комбинируя фреймы и логику первого порядка (First Order Logic — FOL). Однако это еще не означает, что они могут представлять одни и те же компоненты с одним и тем же количеством информации. Только два из перечисленных инструментов, OilEd и OntoSaurus, основаны на дескриптивной логике.

Далее остановимся на некоторых свойствах редакторов онтологий. Интерфейс пользователя редакторов онтологий может быть Web-приложением на основе форм HTML (Ontolingua, OntoSaurus и WebODE) и/или Java-апплетов (WebOnto) или локальным приложением (Protégé, OntoEdit, OilEd).

Все редакторы онтологий за исключением OilEd, Ontolingua и OntoSaurus обеспечивают графические средства редактирования и просмотра онтологий, где классы обычно представлены узлами на графах, а отношения — дугами между ними. Дополнительно к этим графическим функциям, OilEd, OntoEdit Professional, Protégé и WebODE предоставляют некоторую поддержку в написании формальных аксиом и сложных выражений.

OntoEdit, Ontolingua, OntoSaurus, WebODE и WebOnto поддерживают совместную разработку онтологий, предоставляя отдельным пользователям или группам пользователей разрешение на доступ и написание различных наборов онтологий.

Разнообразие инструментов для отображения, выравнивания и объединения онтологий делает сложным их непосредственное сравнение. Фактически, когда разработчик должен решить вопрос, какой инструмент является наиболее подходящим, все будет зависеть от конкретной задачи. Например, если объединяемые онтологии совместно используют набор экземпляров, то лучше всех может работать FCA-Merge. Если онтологии имеют экземпляры, но совместно их не используют, и многие значения слотов содержат текст, лучшим выбором может стать GLUE. Если только части

онтологий должны быть отображены, можно было бы выбрать инструмент ONION. Если онтологии имеют очень ограниченную структуру, а концепты имеют подробные определения на естественном языке (одном), инструментальные средства ISI/USC могут обеспечивать лучшие ответы. Если экземпляры вообще не доступны, и онтологии содержат много отношений между концептами, лучше всех может работать Prompt.

Знания — совокупность сведений, отчетов, фактов, понятий, представлений о чем-либо, накопленных в результате обучения, опыта, в процессе деятельности. Корпоративные знания — знания, которые доступны организации в явном виде и могут использоваться для повышения эффективности сотрудниками данной организации.

Управление знаниями (Knowledge Management) — совокупность процессов и технологий, предназначенных для выявления, создания, распространения, обработки, хранения и предоставления для использования знаний. Управление знаниями — это стратегия предприятия, цель которой — выявить и обратить на пользу фирме всю имеющуюся у нее информацию, опыт и квалификацию сотрудников с тем, чтобы повысить качество обслуживания клиентов и сократить время реакции на меняющиеся рыночные условия. Термин "управление знаниями" начал использоваться еще в середине 1990-х годов в связи с проблемами, возникшими при обработке больших объемов информации в крупных корпорациях. Он связан с поддержкой процессов создания, распространения, обработки и использования знаний внутри предприятия. При этом знания классифицируются и распределяются по категориям в соответствии с предопределенной, но развивающейся онтологиейструктурированных и слабо структурированных баз данных и баз знаний.

Система управления знаниями (СУЗ или Organizational Memory Information Systems — OMIS) является корпоративной информационной системой, предназначенной для хранения, генерирования и доставки пользователям полезной информации по вопросам деятельности компании. Назначение OMIS — накопление информации, позволяющей решать производственные задачи, обеспечение доступности и повторной используемости знаний на уровне всей корпорации. Для этого OMIS должна предоставлять нужные данные, не полагаясь на запросы пользователей. Система должна действовать как интеллектуальный помощник пользователя. В СУЗ знаниями считают всю доступную информацию (документы, сведения о заказчиках, описание технологий работы, продукции и т. д.), а также

закономерности предметной области, полученные из практического опыта или внешних источников.

Одними из первых СУЗ были хранилища данных. В дальнейшем идея хранилища трансформировалась в понятие корпоративной памяти, которая содержит гетерогенную информацию из различных источников и обеспечивает доступ к ней для решения производственных задач.

При построении систем управления знаниями используют такие разделы искусственного интеллекта, как онтологии, многоагентные системы, экспертные системы. OMIS должны быть связаны с другими компонентами корпоративных информационных систем, поддерживающими управление документами и документооборотом.

На физическом уровне СУЗ рекомендуется создавать в виде хранилищ данных, отличающихся от обычных распределенных БД согласованностью хранимой информации.

Архитектура СУЗ и используемые в СУЗ методы функционирования существенно зависят от характера обслуживаемой информации.

В случае сильно структурированных данных управление данными реализуется с помощью обычных средств СУБД. Интеллектуальный анализ данных выполняется методами, характерными для систем OLAP, извлечение сведений о зависимостях и закономерностях, имеющихся в данных, осуществляется с помощью систем Data Mining.

Основу большинства технологий управления слабо структурированными данными составляют модели онтологий и их представление в общем случае в виде тезаурусов и семантических сетей, а в некоторых более частных случаях — в виде морфологических таблиц, альтернативных И-ИЛИ-графов, фреймов, описывающих иерархии целей, показателей, альтернатив и т.п. Применяются также технологии информационно-поисковых систем.

При разработке СУЗ выделяют следующие этапы:

накопление — стихийное и бессистемное накопление информации в организации;

извлечение — процесс, идентичный традиционному извлечению знаний для экспертной системы (один из наиболее сложных и трудоемких этапов, от его успешности зависит дальнейшая жизнеспособность системы);

структурирование — на этом этапе должны быть выделены основные понятия, выработана структура представления информации, обладающая максимальной наглядностью, простотой изменения и дополнения;

формализация — представление структурированной информации в форматах машинной обработки, то есть на языках описания данных и знаний;

обслуживание — под процессом обслуживания понимается корректировка формализованных данных и знаний (добавление, обновление), удаление устаревшей информации, фильтрация данных и знаний для поиска информации, необходимой пользователям.

Примером СУЗ с использованием фреймового представления знаний может служить система, описанная в. В этой системе хранятся сведения о принятых в процессе деятельности компании решениях в области управления качеством с помощью метода анализа иерархий. Перечни альтернатив использовавшихся целей, подцелей и показателей качества, принимавшиеся экспертные оценки и итоговые решения составляют фреймовую структуру. Для доступа к фреймам пользователь должен указать значения параметров, определяющих область (гиперкуб) знаний. Координатными осями гиперкуба приняты такие параметры, как вид продукции, компании-поставщики и компании-потребители, процессы, даты решения проблем и др.

Примером системы, реализующей онтологический подход к представлению и управлению знаниями, может служить система БиГОР. Эта система предназначена для компиляции новых учебных пособий из элементов — отдельных документов (модулей) базы знаний. Очевидно, что в системе могут формироваться любые ассоциации документов, связанных отношениями онтологии, для решения конкретных возникающих задач. Тем самым, БиГОР можно считать системой генерации новых знаний.

Онтологичекая система управления знаниями (онтологическая СУЗ) основана на использовании онтологий. Создаваемая для СУЗ онтология приложения представляется в видесемантической сети и словаря понятий (тезауруса).

Знания в онтологической системе управления знаниями выражены отдельными понятиями (сущностями, концептами) и отношениями между понятиями, которые в семантической сети изображаются соответственно вершинами и ребрами. Каждому понятию могут соответствовать модули знаний (документы, статьи), содержащие необходимые описания, относящиеся к понятию. Модули имеют метаданные, имена понятий входят в число метаданных. Понятия, используемые для определения i-го понятия с именем M_i, называются предшествующими понятиями (по отношению к M_i), их множество обозначим $\mathbf{Q}_i$, а понятия, для определения которых используется M_i, называются последующими, их множество обозначим $\mathbf{G}_i$.

Следовательно, i-й вершине соответствует отношение ($\mathbf{Q}_i$, M_i, $\mathbf{G}_i$), причем для корневой вершины сети $\mathbf{Q}_i = \varnothing$, для терминальных вершин $\mathbf{G}_i = \varnothing$.

Понятия (вершины) можно подразделять на частные (факты) и базовые (знания, обобщающие факты). Создание базовой подсети или ее корректировка и являются собственно генерацией новых знаний. Примером знаний в базовой подсети могут быть методы решения проблем, а фактами в частной подсети — программы решения задач или документы с описаниями действий в конкретных ситуациях, которые возникали в процессе функционирования предприятия (рис. 1).

Рис. 1. Примеры компонентов семантической сети

Общая семантическая сеть в онтологической СУЗ может быть разделена на ряд фрагментов, соответствующих разделам корпоративных знаний. Такими фрагментами могут быть "маркетинг", "кадры", "обеспечение качества продукции" и др. На рис. 2 приведен упрощенный пример фрагмента семантической сети для решения задач реинжиниринга. На рисунке цветными показаны вершины, соответствующие базовой подсети, а неокрашенные вершины соответствуют фактам.

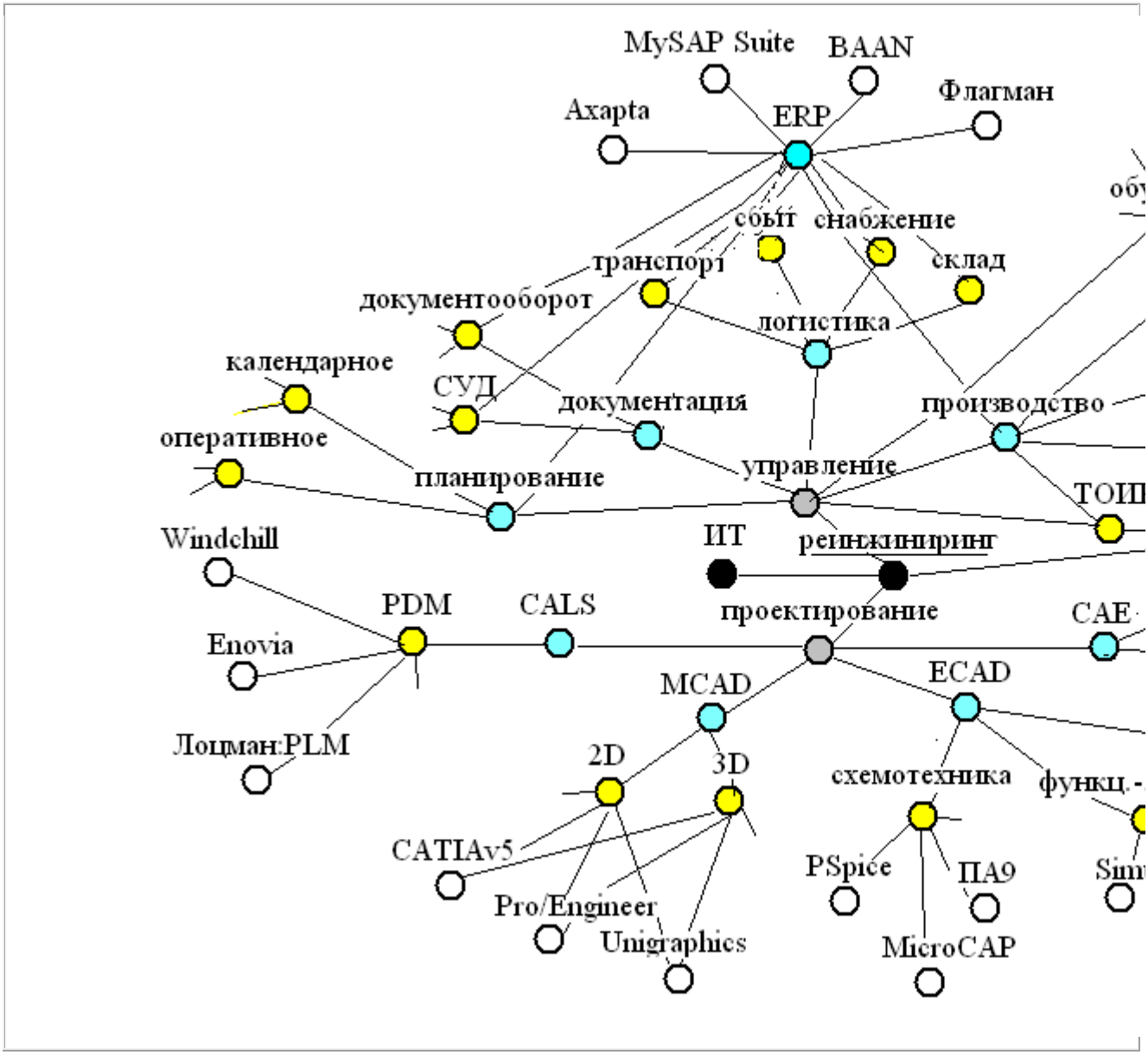


Рис. 2. Фрагмент семантической сети "Реинжиниринг"

Рассмотрим, каким образом могут решаться задачи управления знаниями с помощью онтологической СУЗ.

1. **Задача поиска документов**. Словарь СУЗ содержит все используемые в онтологии понятия. Поиск документов осуществляется с помощью индекса, включающего имя понятия (учитываются только основы слов) и месторасположение относящихся к нему документов в информационной сети.

Особенностью СУЗ является целесообразность реализации методов *поиска по динамически формируемым запросам*. Часто нужный документ характеризуется не концептами, содержащимися в запросе, а близкими к ним в семантическом смысле терминами. Наличие семантической сети (СС) позволяет автоматически расширить поисковый запрос синонимами и

концептами, находящимися в СС в "окрестности" заданного исходного концепта.

2. **Задача принятия решений**. Принятие решений в онтологической СУЗ осуществляется с помощью известного метода поиска в глубину. Автоматический поиск решений возможен, если известна функция оценки формируемых решений — целевая функция, заданная на множестве понятий. Тогда используются те или иные методы дискретной оптимизации, например, генетические алгоритмы. Как правило, такая функция не задана, вследствие чего оценка промежуточных и окончательных решений производится *ЛПР* (лицом, принимающим решение) в интерактивном режиме. Роль СУЗ при этом заключается в прокладывании маршрута поиска в семантической сети с учетом допустимости шагов формирования решения. Очередной шаг поиска заключается в переходе от текущей вершины M_i к одной из вершин $M_j \in \mathbf{G}_i$, после чего M_j становится текущей вершиной. Любая текущая вершина либо принимается ЛПР, либо отвергается. Для этого могут быть использованы экспертные оценки, например, в соответствии с методом анализа иерархий. Далее осуществляется либо переход в новую текущую вершину и ее анализ, либо бектрекинг. Процесс продолжается, пока не будет принято решение о прекращении поиска с фиксацией полученного результата.

Обычно формирование решения происходит при наличии тех или иных ограничений. Типичными ограничениями являются ограничения на совместимость или на принадлежность компонент решения к одной и той же группе (например, выбираемых единиц оборудования к продукции одного и того же производителя). Эти ограничения учитываются в семантической сети включением в сеть некоторых дополнительных вершин (например, вершин производителей оборудования) и соответствующих ребер.

3. **Генерация новых знаний**. Генерацию новых знаний следует отождествлять либо с выделением в уже сформированной базе знаний компонентов, отвечающих условиям поставленной задачи, либо с включением в базовую часть семантической сети новых элементов, каковыми могут быть:

- новые понятия;
- новые отношения между понятиями;
- новые отношения между понятиями и документами.

Выделение компонентов в имеющейся базе знаний в простейшем случае совпадает с задачей поиска документов по заданным ключевым словам. В случаях, когда результат поиска на множестве документов может быть оценен количественно, задача сводится к задаче принятия решений.

Включение новых элементов в семантическую сеть может осуществлять ЛПР (администратор базы знаний). Но возможны и полуавтоматические методы генерации. Так, выявление новых концептов — это поиск специфических терминов в новых документах, поступающих в БД. Такими терминами являются термины, отсутствовавшие в СС и встречающиеся только в новом документе.

Новые отношения могут устанавливаться по признаку совместного использования концептов при решении конкретной проблемы. Это динамически определяемые отношения, причем ослабевающие со временем (для них целесообразно установить некоторый "период полураспада").

Новые отношения между понятиями и документами могут включаться в СС, если в документах СУЗ автоматически выделяются все термины (как в БиГОРе), соответствующие понятиям онтологии. Пользователю остается лишь отобрать нужные термины, которые будут трансформироваться в гиперссылки. Новые отношения между понятиями и документами появляются также при добавлении в метаданные документов ссылок на термины.

Реализация онтологической СУЗ возможна с помощью онтологических систем формирования учебных материалов типа БиГОР. Для моделирования онтологической СУЗ, помимо семантической сети, можно использовать сети Петри или диаграммы деятельности языка UML. С их помощью отображается процесс принятия решения как последовательность событий.

ТЕМА №3. Семантический Web. Метаданные. Модель метаданных RDF. Язык RDFS.

Семантический Web представляет собой иерархическую структуру, включающую несколько слоев моделей и языков описания информации (рис. 1).

В основании иерархической пирамиды находятся способы кодирования (форматы) данных, такие как, например, ASCII, UNICODE, графические форматы, а также ссылочные идентификаторы URI(Universal Resource Identifier).

Выше расположен уровень языков разметки HTML и XML (вместе со схемами HTML и XML или таблицами определения типов DTD).

Далее следует уровень с моделью RDF и языком RDFS (RDF Schema), служащими для описания метаданных, их интерпретации и обмена данными между приложениями. Если на уровне языков HTML и XML рассматриваются вопросы, связанные только со структурой документов, то на уровне RDF/RDFS решаются проблемы обеспечения семантической интероперабельности документов.

Верхний уровень занимают модели и языки онтологии, создаваемые на базе RDF/RDFS-средств. К языкам онтологии относятся DAML (DARPA Agent Markup Language), OIL (Ontology Interchange Language), OWL (Web Ontology Language) и др. С их помощью можно устанавливать эквивалентность классов, представлять теоретико-множественные операции над классами и т.п.

Применение семантического Web направлено на повышение эффективности решения следующих проблем:

- расширенная навигация в информационном Web-пространстве и многомерный поиск;
- семантическая интероперабельность порталов и других источников и хранилищ информации; данные из разных источников и разных форматов могут быть интегрированы в одном приложении;
- реструктуризация информации в порталах, описание содержимого и взаимосвязей веб-сайтов, страниц, библиотек.

Следует отметить, что для всех языков разметки на верхних иерархических уровнях, представленных на рис. 1, в качестве основы принят язык XML.

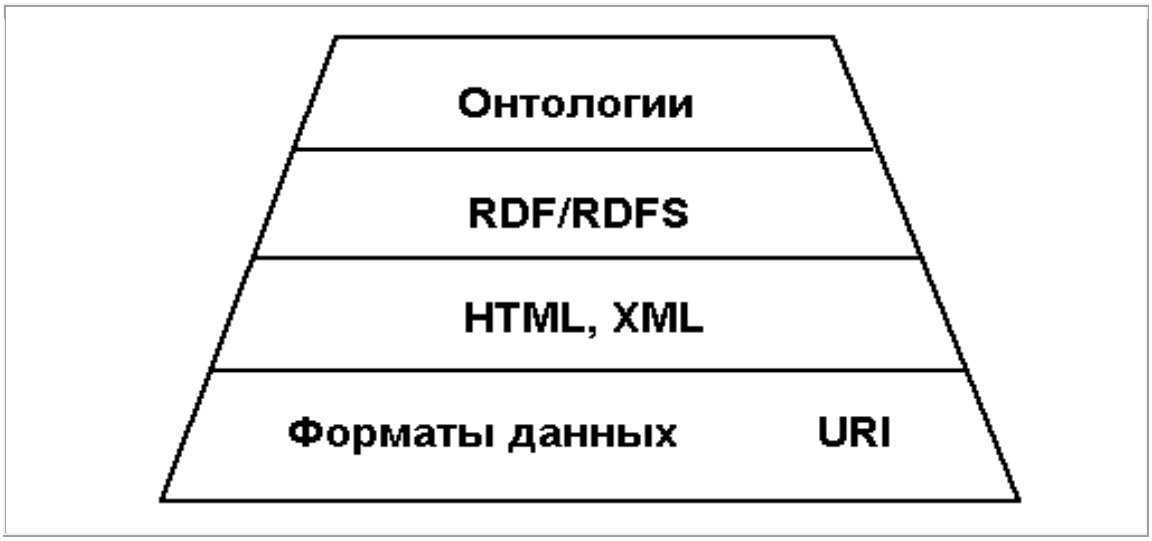


Рис. 1. Семантический Web

Метаданными принято называть данные о данных. Так, в системах документооборота, хранения и поиска документов метаданные называют поисковым образом. В этих системах документы должны помимо содержательной части иметь некоторые атрибуты, характеризующие те или иные свойства документов. Атрибуты полезно используются для классификации, отбора и поиска документов.

Метаданные часто представляют собой некоторую фреймовую структуру, с определенным числом слотов (полей) и их назначением.

Существует ряд унифицированных систем представления метаданных. К ним относятся системы:

- Dublin Core Metadata Element Set (Дублинское ядро), для различных информационных ресурсов и получившая распространение в библиотечных системах;
- GEM, являющаяся частным случаем Дублинского ядра для образовательных ресурсов;
- LOM (Learning Object Metadata), разработанная в IEEE и принятая за основу организацией IMS для описания метаданных образовательных ресурсов.

В Web-технологиях для описания метаданных различных документов предложены модель RDF и язык RDFS (RDF Schema), используемые в процедурах интерпретации данных и обмена данными между приложениями.

RDF (Resource Description Framework) — модель, предложенная консорциумом World Wide Web Consortium (W3C) для описания метаданных информационных ресуросов в Web. Для представления модели RDF используется язык XML. В получающемся XML-документе RDF-метаданные помещаются в контейнер XML с дескриптором rdf:RDF.

Модель состоит из предложений вида **субъект - предикат - объект**, где субъект — ресурс, о котором идет речь в предложении, предикат — ресурс, являющийся свойством субъекта, объект — ресурс, являющийся значением свойства. Ресурсами являются некоторые модельные представления об объектах реального мира, обычно описываемые в документах, имеющих URI. Отметим, что свойство в RDF — понятие, которое может использоваться самостоятельно, вне связи с субъектом. Пример предложения:

Команда имеет тренера, его имя Иванов.

Здесь субъектом, предикатом и объектом являются команда, тренер, Иванов соответственно.

Ресурсы могут объединяться в группы, называемые классами. Сами классы также являются ресурсами и идентифицируются ссылками RDF-URI. В RDF определение класса или свойства (так называемый интенсионал) отделено от множества экземпляров класса и значений свойства (от экстенсионала).

Для записи RDF-модели используют язык XML. RDF-модель документа на языке XML есть последовательность элементов Description. Каждый элемент Description применяется к одному определенному ресурсу и содержит описания свойств ресурса (т.е. содержит значение определенного поля метаданных документа). URI этого ресурса указывается в Description в качестве атрибута about или ID. Например:

<rdf:Description about="http://...">

Свойство может быть URI, строкой символов или элементом Description. Возможно использование нескольких элементов Description как включаемых в модель последовательно, так и вложенных друг в друга. Свойства представляются в виде XML-элементов <предикат>объект</предикат> или <предикат rdf:resource="объект"/>.

Общая структура модели имеет вид:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="субъект">
  <предикат>объект</предикат>
</rdf:Description>
  ...
<rdf:Description rdf:about=...>
  <предикат rdf:resource="объект"/>
</rdf:Description>
</rdf:RDF>
```

Поскольку модель RDF относится к описанию семантики предметной области и служит, в частности, для согласования семантики документов из разных источников, необходимо решать проблемы возможного использования в этих документах терминов-омонимов. Проблемы омонимии решаются с помощью использования пространств имен.

Предположим, что некоторый термин "system" имеет разный смысл в предметных областях, описываемых в документах с URI http://www.cl.edu/cont/ и http://aba.ru/technologies/elements/. Тогда на этот термин в RDF-модели нужно ссылаться с помощью составных имен (QNames), например, m1:system и m2:system, причем предварительно должны быть введены псевдонимы m1 и m2 для указанных URI. Структура RDF-модели может иметь вид (в примере использование m1:system и m2:system подразумевается в XML-предложениях внутри элементов rdf:Description):

```
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:m1="http://www.cl.edu/cont/"
      xmlns:m2="http://aba.ru/technologies/elements/">
<rdf:Description rdf:about=...>
  ...
</rdf:Description>
</rdf:RDF>
```

Приведем простой пример RDF-модели:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <rdf:Description rdf:about="http://www.work.ru/person/">
    <contact:fullName>Ivan Ivanov</contact:fullName>
    <PlaceOfBirth>Moscow</PlaceOfBirth>
    <contact:personalTitle>Professor</contact:personalTitle>
  </rdf:Description>
</rdf:RDF>
```

Здесь элемент rdf:Description служит для представления метаданных документа, находящегося по адресу http://www.work.ru/person/, другими словами, .http://www.work.ru/person/ есть URI субъекта, его свойствами являются полное имя, место рождения, звание, а значениями свойств соответственно Иван Иванов, Москва и профессор.

Полные URI можно заменять псевдонимами, используя объявления ENTITY в таблице определения типов DTD. Например:

<!ENTITY dev "http://ee.com/product/comp#">

Теперь полный URI (http://ee.com/product/comp#diode) может быть заменен на &dev;diode.

Язык *RDFS* — язык описания словарей классов и свойств Web-ресурсов, предназначен для определения, стандартизации и использования метаданных, описывающих ресурсы Web. Так же какXML схема служит для задания структуры и типов в документах XML, язык RDFS нужен для описания структуры и типов для моделей RDF.

Язык RDFS предоставляет базовую систему типов для моделей RDF (аналогично Express в STEP). В числе типов имеются некоторые предопределенные типы, такие как **Class**, **subPropertyOf** и**subClassOf**, используемые всеми интегрируемыми приложениями. В RDFS вводятся понятия класса, подкласса, свойства и подсвойства, имеется возможность накладывать на них ограничения. В модели RDF используется иерархическое построение классов (сущностей) и свойств (атрибутов) предметной области, определяется, какие свойства и с какими классами могут быть использованы. Концепция RDF близка к концепции семантической сети.

Как и в XML, в RDFS вводятся теги для задания типов и атрибутов метаданных.

Множество ресурсов, родственных в том или ином смысле, является классом. Классы задаются с помощью тега **rdfs:Class**, их принадлежность к некоторому надклассу описываются с помощью тега **rdfs:Resource, т.е. данный класс является ресурсом такого-то надкласса**. К числу классов относится **rdfs:Literal**, его экземплярами могут быть, например,**string** и **integer**. Экземплярами класса **rdfs:Datatype**, т.е. типов данных, являются подклассы класса **rdfs:Literal**. Принадлежность ресурса к конкретному классу задается с помощью свойства **rdf:type**.

Класс **rdf:Property** служит для описания свойств ресурсов. Свойство может быть выражено литералом или быть отношением двух классов. Класс свойств **rdfs:domain** позволяет указать область определения свойства, т.е. классы, к которым это свойство может быть применено. Класс **rdfs:range** позволяет указать диапазон свойства, т.е. классы, чьи ресурсы могут появиться в качестве значений заданного свойства.

Свойства **rdfs:subClassOf** и **rdfs:subPropertyOf** относятся соответственно к подклассу класса и подсвойству свойства.

Например, описание класса diode как подкласса класса device со свойством material может иметь следующий вид:

<rdfs:Class ID="diode">

```
  <rdfs:subClassOf rdf:resource="#device"/>
 </rdfs:Class>
 <rdf:Property ID="material">
  <rdfs:domain rdf:resource="#diode"/>
  <rdfs:range rdf:resource="#string"/>
 </rdf:Property>
```

Свойства **rdfs:label** и **rdfs:comment** применяются для задания удобных для человека псевдонимов именам и содержимому ресурсов.

В таблице 1 перечислены классы RDFS, а в таблице 2 — свойства RDFS.

Таблица 1

Имя класса	**Пояснение**
rdfs:Resource	Класс-ресурс, включает "всё"
rdfs:Literal	Класс литеральных значений, текстовых строк или чисел
rdf:XMLLiteral	Класс XML-литералов
rdfs:Class	Класс классов
rdf:Property	Класс RDF-свойств
rdfs:Datatype	Класс типов данных RDF
rdf:Statement	Класс утверждений
rdf:Bag	Класс неупорядоченных контейнеров
rdf:Seq	Класс упорядоченных контейнеров
rdf:Alt	Класс контейнеров-альтернатив
rdfs:Container	Класс RDF-контейнеров
rdfs:ContainerMembership	Класс свойств "членства" в контейнерах
rdf:List	Класс RDF-списков

Индивидами являются сущности, состоящие из единственного объекта.

Описание типа rdf:type служит для связывания индивида с классом, членом которого он является. Например, в следующем предложении X является экземпляром класса P:

<owl:Thing rdf:about="#X">
<rdf:type rdf:resource="#P"/>
</owl:Thing>

Но то же можно выразить и таким образом:

<P rdf:ID="X" />

Таблица 2

Имя свойства	Пояснение	Домен	Диапазон
rdf:type	Субъект является экземпляром класса	rdfs:Resource	rdfs:Class
rdfs:subClassOf	Субъект является подклассом класса	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	Субъект является подсвойством свойства	rdf:Property	rdf:Property
rdfs:domain	Домен свойства субъекта	rdf:Property	rdfs:Class
rdfs:range	Диапазон свойства субъекта	rdf:Property	rdfs:Class
rdfs:label	Человекочитаемое название субъекта	rdfs:Resource	rdfs:Literal
rdfs:comment	Текстовое описание ресурса	rdfs:Resource	rdfs:Literal
rdfs:member	Член ресурса субъекта	rdfs:Resource	rdfs:Resource
rdf:first	Первый элемент списка	rdf:List	rdfs:Resource
rdf:rest	Оставшийся за первым элементом "хвост" списка	rdf:List	rdf:List
rdfs:seeAlso	Дополнительная информация о субъекте	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	Определение ресурса субъекта	rdfs:Resource	rdfs:Resource
rdf:value	Свойство, используемое для структурированных значений	rdfs:Resource	rdfs:Resource
rdf:subject	Субъект RDF-утверждения (см.	rdf:Statement	rdfs:Resource

	"реификация")		
rdf:predicate	Предикат RDF-утверждения (см. "реификация")	rdf:Statement	rdfs:Resource
rdf:object	Объект RDF-утверждения (см. "реификация")	rdf:Statement	rdfs:Resource

ТЕМА №4. Дублинское ядро. Языки онтологий. Язык OWL. Web-2.

Наиболее перспективным средством формирования описательных метаданных для широкого класса цифровых объектов является, по мнению многих ученых, система метаданных Дублинского ядра DC (*Dublin Core*).

Система Дублинского ядра разрабатывается с 1995 г.

Набор метаданных Дублинского ядра состоит из 15 элементов. Все элементы являются необязательными.

Ниже приводится список элементов DC, взятый из книги Вильяма Армса «Электронные библиотеки».

- Title (Заголовок) — название, присвоенное ресурсу создателем или издателем.
- Creator (Автор) — человек или организация, имеющие непосредственное отношение к созданию содержимого ресурса (например, авторы; фотографы, иллюстраторы и т.п.).
- Subject (Предмет) — тема ресурса. Обычно предмет выражается в ключевых словах или фразе, описывающей предмет или содержание ресурса.
- Description (Описание) — текстовое описание содержания ресурса, включая реферат в случае документов или описание содержания в случае визуального ресурса.
- Publisher (Издатель) — организация, ответственная за создание ресурса в его нынешней форме — например, издательство, университетский департамент или корпорация.
- Contributor (Участник создания материала) — человек или организация, которые не являются авторами (не обозначены в элементе «автор»), но внесли значительный интеллектуальный вклад в ресурс, но чья роль менее значима, чем авторов (например, редактор или переводчик).
- Date (Дата) — дата, указывающая на создание или появление (в доступном виде) ресурса.
- Type (Тип) — категория ресурса (например, учебник, поэма, статья, справочник, технический отчет, словарь).
- Format (Формат) — формат представления данных ресурса (например, doc, xml, ftp).

- Identifier (Идентификатор) — набор букв или цифр, который обычно используется для уникальной идентификации ресурса. В случае сетевых ресурсов примером является URL.
- Source (Источник) — информация о вторичном источнике, из которого был получен настоящий ресурс.
- Language (Язык) — язык, на котором изложено содержание ресурса.
- Relation (Связь) — идентификатор вторичного ресурса и его связь с настоящим ресурсом. Этот элемент позволяет связывать между собой близкие ресурсы, а также описания ресурса, которые необходимо показать. Примеры — издание книги и глава книги.
- Coverage (Охват) — характеристики местонахождения и временной продолжительности ресурса.
- Rights (Права) — утверждение об авторских правах и управление ими; идентификатор, связанный с таким утверждением; идентификатор, связанный с сервисом, представляющим информацию об управлении правами на данный ресурс.

Согласно рекомендации RFC 2413, все элементы Дублинского ядра, описанные в перечне, можно разбить на три группы:

- элементы, относящиеся к содержанию ресурса (Content);
- элементы, описывающие цифровой ресурс с точки зрения интеллектуальной собственности (Intellectual Property);
- элементы, относящиеся к конкретному экземпляру ресурса (Instanitation).

В нижеприведенной таблице эта группировка раскрыта более полно:

Таблица 1

Content	Intellectual Property	Instantiation
Title, Subject, Description, Type, Source, Relation, Coverage	Creator, Publisher, Contributor, Rights	Date, Format, Identifier, Language

Примеры использования метаданных Дублинского ядра, встроенных в HTML и RDF документы (взяты из http://xmlhack.ru/archives/2005/06/000111.html):

<meta name="DC.subject" content = "dublin core metadata element set">

<meta name="DC.subject" content = "networked object description">

<meta name="DC.publisher" content="OCLC Online Computer Library Center Inc."

<meta name="DC.creator" content="Weibel, Stuart L., wei...@oclc.org.">

<meta name="DC.creator" content="Miller, Eric J., emil...@oclc.org.">

<meta name="DC.title" content="Dublin Core Element Set Reference Page">

<meta name="DC.date" content="1996-05-28">

<meta name="DC.form" scheme="IMT" content="text/html">

<meta name="DC.language" scheme="ISO 639" content="en">

<meta name="DC.identifier" scheme="URL"
content="http://purl.oclc.org/metadata/dublin_core">

Запись в формате RDF:

<RDF:RDF>

<RDF:description RDF:about= "http://hamlet.org">

<DC:creator> Shakespeare </DC:creator>

<DC:type> play </DC:type>

</RDF:description>

</RDF:RDF>

Онтологии являются описаниями семантики конкретных предметных областей в виде совокупности сущностей, их атрибутов, значений атрибутов и ограничений. В свою очередь, сущностьопределяется как множество однородных объектов или как некоторое понятие предметной области. Множество сущностей можно представить в виде словаря (тезауруса) понятий.

Одним из развитых *языков онтологий* является язык Express, созданный для целей информационной поддержки промышленных изделий на различных этапах их жизненного цикла и изложенный в группе стандартов STEP (ISO 10303). Онтологии ряда приложений, выраженные на языке Express и представленные в прикладных протоколах STEP, используются для создания электронных макетов изделий и информационного согласования различных промышленных автоматизированных систем.

Однако язык Express не приспособлен для использования в таких приложениях, как, например, образование, он не предназначен для описания текстовых документов. Широкое применение языка разметки XML для структурирования текстовых документов, в том числе учебных материалов, обусловливает целесообразность описания онтологий предметных областей в информационно-образовательных средах средствами языков, базирующихся на XML.

Шагом на пути перехода от языка XML к языкам онтологий стало создание модели и языка описания обмена метаданными RDF/RDFS, разработанных практически одновременно с XML. Язык RDFSможно рассматривать как надстройку над XML. Отметим, что обычно метаданные интерпретируют как данные о данных, а в базах знаний под метаданными подразумевают синтаксические и семантические правила обработки информации.

Язык RDFS может служить языком общения программных систем, работающих в среде Internet, на его основе происходит определение, стандартизация и использование метаданных, описывающих ресурсы Web. С помощью RDFS решается проблема поиска ресурса по его свойствам, могут быть получены сведения о том, как документы связаны друг с другом, где брать описание типов документов и т.д.

Собственно языки онтологий для Web создаются на базе RDF/RDFS. Так, язык онтологий OIL является расширением RDFS, в котором используются фреймовые представления из моделей искусственного интеллекта.

Язык онтологий DARPA Agent Markup Language (DAML) предложен в 2000 г. Целью DAML и реализующего его программного обеспечения являются динамическая идентификация и интерпретация данных, интероперабельность между программными агентами, представление метаданных и управление знаниями. Объединение языков DARPA и OIL расширило возможности классификации, типизации и описания свойств ресурсов.

Другой язык онтологий *OWL* (Ontology Web Language), созданный на базе XML, также является элементом стека web-протоколов. В нем используется объектно-ориентированный метод описания предметных областей, реализуется представление о приложении, как о множестве сущностей (объектов) с наборами ограничений и атрибутов, характеризующих свойства и межобъектные связи. В языке есть средства описания версий онтологий и их агрегирования. Предполагается наличие агентов, которые смогут автоматически пополнять базу знаний информацией, вновь появляющейся в Интернет, строить тестирующие системы и т.п.

Примерами редакторов для разработки онтологий могут служить программы Webonto или OilEd (последняя поддерживает языки OIL и RDFS).

Существующие языки онтологий являются декларативными. Основная область применения таких языков, как RDF/RDFS, OIL, DAML, OWL - информационный поиск. Имеются языки, ориентированные на создание экспертных систем, например LOOM. С момента их создания первых языков онтологий прошло пока еще мало времени. Следует ожидать появления более

мощных языков, позволяющих описывать как структурные, так и поведенческие аспекты приложений.

В языках онтологий должны быть средства описания классов (концептов) предметной области, экземпляров классов, свойств концептов и отношений между концептами. Рассмотрим, как эти части онтологий описываются в языке *OWL* (WEB Ontology Language).

Задание класса X осуществляется следующим образом:

<owl:Class rdf:ID="X"/>

Далее может использоваться ссылка на класс X в виде #X.

Экземпляр Y класса X (индивид) может быть задан в виде

<X rdf:ID="Y" />

Свойствами могут быть ObjectProperty, DatatypeProperty, rdfs:subPropertyOf, rdfs:domain, rdfs:range.

Например, свойство A, относящееся к концептам X и Z (ограничения domain и range — см. язык RDFS), будет выглядить таким образом

<owl:ObjectProperty rdf:ID="A">
<rdfs:domain rdf:resource="#X"/>
<rdfs:range rdf:resource="#Z"/>
</owl:ObjectProperty>

Отношения иерархии классов и свойств выражаются конструкциями subClassOf и subPropertyOf. Например, пусть Z — подкласс класса X. Тогда

<owl:Class rdf:ID="Z">
<rdfs:subClassOf rdf:resource=#X" />
</owl:Class>

Отношения представляются как свойства. Например, отношение A между Y1 и Y2 и отношение C между Y1 и Y3 можно описать так

<X rdf:ID="Y1">
<имя свойства A rdf:resource="#Y2" />
<имя свойства C rdf:resource="#Y3" />
</X>

Отношения могут быть симметричными, несимметричными, функциональными, транзитивными и т.п., что в случае, например, симметричного отношения может быть выражено строкой <rdf:type rdf:resource="&owl;SymmetricProperty" /> в

<owl:ObjectProperty rdf:ID="A">
<rdf:type rdf:resource="&owl;SymmetricProperty" />
. . .
</owl:ObjectProperty>

Если свойство A имеет подсвойство C, то

<owl:ObjectProperty rdf:ID="C">

<rdfs:subPropertyOf rdf:resource="#A" />

</owl:ObjectProperty>

В язык онтологий OWL дополнительно по отношению к RDFS введены такие отношения классов и свойств, как **equivalentClass**, **equivalentProperty**, **sameAs** (один и тот же),**differentFrom**, **AllDifferent**, **distinctMembers**.и др.

Средства **equivalentClass** и **equivalentProperty** используются для задания отношения эквивалентности между двумя классами или двумя свойствами. Средство **sameAs** позволяет создать несколько имен для одного и того же объекта. С помощью **differentFrom** можно задать отличие одного ресурса от других, а с помощью **AllDifferent** указать, что все перечисленные экземпляры различны.

ObjectProperty — используемое в OWL средство для описания отношения объектов. При описании свойств используются характеристики **inverseOf** (инверсия свойства),**FunctionalProperty**, **InverseFunctionalProperty** (инверсия функциональная), **TransitiveProperty**, **SymmetricProperty** и ограничения **AllValuesFrom** и **SomeValuesFrom**.

Примерами инверсных свойств, описываемых с помощью **inverseOf**, могут служить больше-меньше, родитель-потомок, холоднее-жарче и т.п. В функциональных отношениях и при функциональной инверсии определяемое свойство может не иметь ни одного или иметь только одно значение. Так, для субъекта "человек" свойство "место жительства" является функциональным. Если пара (x,y) объявлена с помощью **TransitiveProperty** как транзитивное отношение P и пара (y,z) — экземпляр того же отношения P, то пара (x,z) также экземпляр отношения P. Так, отношение "больше" — транзитивное в отличие, например, от отношения "отец-сын". Отношение симметричности задается с помощью характеристики **SymmetricProperty**, примером такого отношения может служить отношение партнерства.

Ограничение **AllValuesFrom** для некоторого свойства указывает класс, к которому должны принадлежать значения этого свойства. В случае **SomeValuesFrom** хотя бы некоторые значения свойства должны принадлежать указанному классу.

Пример 1

Отношение эквивалентности классов paper и publication и их отличие от класса letter:

<owl:Class rdf:about="paper">

<owl:equivalentClass rdf:resource="publication"/>

<owl:differentFrom rdf:resource="letter"/>

</owl:Class>

Пример 2

Описание свойства founded_before_1900, применяемого к классу City и принимающего значения в классе capitals, может быть представлено следующим образом:

<owl:ObjectProperty rdf:ID="founded_before_1900">

<owl:domain rdf:resource="#City"/>

<owl:range rdf:resource="#capitals"/>

</owl:ObjectProperty>

Здесь использована возможность присвоения классам и свойствам имен с помощью атрибута ID, например:

<owl:Class rdf:ID="City"/>

Теперь ссылка на этот класс имеет вид #City.

Web-2 система Web, которая из пространства публичного представления материалов превращает Интернет в пространство публичного формирования и использования материалов. Строится на основе порталов, адаптирующихся к требованиям пользователей. Главный принцип Web-2 — добровольное сотрудничество людей с целью организации источников информации. В Web-2 формируется отношение к Web как к социальной сети, в которой пользователи осуществляют систематизацию контента.

RSS — это формат представления данных и технология автоматического взаимодействия между компьютерами на основе доступа к документам в этом формате с целью сбора информации и представления ее пользователям. Формат RSS образован на базе XML. Технология RSS позволяет компьютерам автоматически распознавать и отбирать информацию, нужную пользователям, составлять списки тем и предметов, интересующих конкретного пользователя, и следить за изменением соответствующих ресурсов. Используется несколько расшифровок аббревиатуры RSS, например RDF Site Summary или Really Simple Syndication. Другим форматом аналогичного назначения является *Atom*. Существенной частью спецификации Atom является протокол, работающий поверх HTTP. Активно поддерживается компанией Google.

Web-сайты, поддерживающие RSS, называются каналами, они регистрируются и являются источниками информации. Доступ к каналу осуществляется так же, как и к любому другому ресурсу или файлу, находящемуся на Web-сервере.

Любой узел сети, снабженный специальной программой, называемой *RSS-агрегатором*, объединителем или считывателем новостей, может обращаться к каналу. Через определенные интервалы времени приложение пользователя автоматически проверяет каналы RSS, выделяет новый материал и автоматически доставлет его в компьютер пользователя. RSS поддерживают почти все блоги и многие новостные сайты. Примеры RSS-агрегаторов: Google Reader, агрегатор Bloglines, Яндекс.Лента. Практически все агрегаторы поддерживают как RSS, так и Atom.

Слово "*блог*" произошло от английского "weblog", что дословно переводится как "веб-записи". Т.е. блог — это персональные заметки, которые публикуются в открытом доступе. Часто блоги называют сетевыми или онлайновыми дневниками.

Другим форматом, того же назначения что и RSS, является Atom.

ТЕМА №5. Эволюционные методы. Простой генетический алгоритм. Кроссовер. Примеры применения генетических методов.

Эволюционные методы (ЭМ) являются приближенными (эвристическими) методами решения задач оптимизации и структурного синтеза. Большинство ЭМ основано на статистическом подходе к исследованию ситуаций и итерационном приближении к искомому решению.

Эволюционные вычисления составляют один из разделов искусственного интеллекта. При построении систем ИИ по данному подходу основное внимание уделяется построению начальной модели, и правилам, по которым она может изменяться (эволюционировать). Причем модель может быть составлена по самым различным методам, например, это может быть и нейронная сеть и набор логических правил. К основным эволюционным методам относятся методы отжига, генетические, поведения "толпы" (PSO), колонии муравьев (ACO), генетического программирования.

В отличие от точных методов математического программирования ЭМ позволяют находить решения, близкие к оптимальным, за приемлемое время, а в отличие от других эвристических методов оптимизации характеризуются существенно меньшей зависимостью от особенностей приложения (т.е. более универсальны) и в большинстве случаев обеспечивают лучшую степень приближения к оптимальному решению. Универсальность ЭМ определяется также применимостью к задачам с неметризуемым пространством управляемых переменных (т.е. среди управляемых переменных могут быть и лингвистические величины, т.е. не имеющие количественного выражения).

В *методе отжига* (Simulated Annealing) имитируется процесс минимизации потенциальной энергии тела во время отжига деталей. В текущей точке поиска происходит изменение некоторых управляемых параметров. Новая точка принимается всегда при улучшении целевой функции и лишь с некоторой вероятностью при ее ухудшении.

Важнейшим частным случаем ЭМ являются *генетические методы* и алгоритмы. Генетические алгоритмы (ГА) основаны на поиске лучших решений с помощью наследования и усиления полезных свойств множества объектов определенного приложения в процессе имитации их эволюции.

Свойства объектов представлены значениями параметров, объединяемых в запись, называемую в ЭМ *хромосомой*. В ГА оперируют подмножеством хромосом, называемом популяцией. Имитация генетических

принципов — вероятностный выбор родителей среди членов популяции, скрещивание их хромосом, отбор потомков для включения в новые поколения объектов на основе оценки целевой функции — ведет к эволюционному улучшению значений целевой функции (функции полезности) от поколения к поколению.

Среди ЭМ находят применение также методы, которые в отличие от ГА оперируют не множеством хромосом, а единственной хромосомой. Так, метод дискретного локального поиска (его англоязычное название Hillclimbing) основан на случайном изменении отдельных параметров (т.е. значений полей в записи или, другими словами, значений генов в хромосоме). Такие изменения называют мутациями. После очередной мутации оценивают значение функции полезности F (Fitness Function) и результат мутации сохраняется в хромосоме только, если F улучшилась. При "моделировании отжига" результат мутации сохраняется с некоторой вероятностью, зависящей от полученного значения F.

В методе PSO (Particles Swarm Optimization) имитируется поведение множества агентов, стремящихся согласовать свое состояние с состоянием наилучшего агента.

Метод колонии муравьев (ACO) основан на имитации поведения муравьев, минимизирующих длину своих маршрутов на пути от муравьиной кучи до источника пищи.

Для применения ГА необходимо:

1. выделить совокупность свойств объекта, характеризуемых внутренними параметрами и влияющих на его полезность, т.е. выделить множество управляемых параметров $\mathbf{X} = (x_1, x_2 \ldots x_n)$; среди x_i могут быть величины различных типов (**real**, **integer**, **Boolean**, **enumeration**). Наличие нечисловых величин (enumeration) обусловливает возможность решения задач не только параметрической, но и структурной оптимизации;
2. сформулировать количественную оценку полезности вариантов объекта — функцию полезности F. Если в исходном виде задача многокритериальна, то такая формулировка означает выбор скалярного (обобщенного) критерия;
3. разработать математическую модель объекта, представляющую собой алгоритм вычисления F для заданного вектора $\mathbf{X}$;
4. представить вектор $\mathbf{X}$ в форме *хромосомы* — записи следующего вида (см. рис. 1).

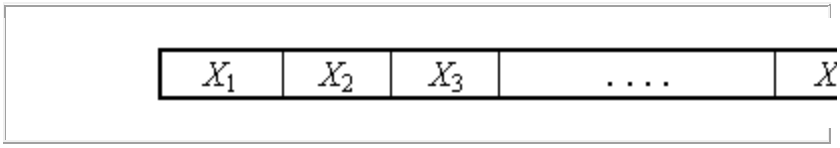


Рис. 1. Хромосома

В ГА используется такая терминология:

- *ген* — управляемый параметр x_i;
- *аллель* — значение гена;
- *локус* (позиция) — позиция, занимаемая геном в хромосоме;
- *генотип* — экземпляр хромосомы, генотип представляет совокупность внутренних параметров проектируемого с помощью ГА объекта;
- *генофонд* — множество всех возможных генотипов;
- функция полезности (приспособленности) F — целевая функция;
- *фенотип* — совокупность значений критериев, получаемых после декодирования хромосомы, под фенотипом часто понимают совокупность выходных параметров синтезируемого с помощью ГА объекта.

Вычислительный процесс начинается с генерации исходного поколения — множества, включающего N хромосом, N — размер популяции. Генерация выполняется случайным выбором аллелей каждого гена.

Далее организуется циклический процесс смены поколений:

```
for (k=0; k<G; k++)
{ for (j=0; j<N; j++)
  { Выбор родительской пары хромосом;
    Кроссовер;
    Мутации;
    Оценка функции полезности F потомков;
    Селекция;
  }
  Замена текущего поколения новым;
}
```

Для каждого витка внешнего цикла генетического алгоритма выполняется внутренний цикл, на котором формируются экземпляры нового (следующего за текущим) поколения. Во внутреннем цикле повторяются операторы выбора родителей, кроссовера родительских хромосом, мутации, оценки приспособленности потомков, селекции хромосом для включения в очередное поколение.

Рассмотрим алгоритмы выполнения операторов в простом генетическом алгоритме.

1. Выбор родителей.

Этот оператор имитирует естественный отбор, если отбор в родительскую пару хромосом с лучшими значениями функции полезности F более вероятен. Например, пусть F требуется минимизировать.

Тогда вероятность P_i выбора родителя с хромосомой C_i можно рассчитать по формуле

$$P_i = \frac{F_{\max} - F_i}{\sum_{j=1}^{N} (F_{\max} - F_j)}, \qquad (1)$$

где $F_{\max}$ — наихудшее значение целевой функции F среди экземпляров (членов) текущего поколения, F_i — значение целевой функции i-го экземпляра.

Правило (1) называют *правилом колеса рулетки*. Если в колесе рулетки выделить секторы, пропорциональные значениям $F_{\max} - F_i$, то вероятности попадания в них суть P_i, определяемые в соответствии с (1).

Пример 1

Пусть $N = 4$, значения F_i и P_i приведены в табл. 1.

Таблица 1

i	F_i	F_1	P_i
1	2	5	0,5
2	7	0	0
3	6	1	0,1
4	3	4	0,4

2. *Кроссовер* (скрещивание).

Кроссовер, иногда называемый кроссинговером, заключается в передаче участков генов от родителей к потомкам. При простом (одноточечном) кроссовере хромосомы родителей разрываются в некоторой позиции, одинаковой для обоих родителей, выбор места разрыва равновероятен, далее происходит рекомбинация образующихся частей родительских хромосом, как это показано в табл. 2, где разрыв подразумевается между пятым и шестым локусами.

Таблица 2

Хро								

мосома	ен 1	ен 2	ен 3	ен 4	ен 5	ен 6	ен 7	ен 8
Родителя A								
Родителя B								
Потомка C								
Потомка D								

3. *Мутации*.

Оператор мутации выполняется с некоторой вероятностью P_m, т.е. с вероятностью P_m происходит замена аллеля случайным значением, выбираемым с равной вероятностью в области определения гена. Именно благодаря мутациям расширяется область генетического поиска.

4. *Селекция*.

После каждого акта генерации пары потомков в новое поколение включается лучший экземпляр пары.

Внутренний цикл заканчивается, когда число экземпляров нового поколения станет равным N. Количество повторений G внешнего цикла чаще всего определяется автоматически по появлению признаков вырождения (стагнации) популяции, но с условием не превышения заданного лимита машинного времени.

Возможны отклонения от представленной в простом генетическом алгоритме (ГА) схемы вычислений.

Во-первых, допустимы схемы многоточечного *кроссовера*. В большинстве литературных источников рассматриваются одноточечные, двухточечные и однородные ГА.

В одноточечном алгоритме при кроссовере каждая из родительских хромосом делится на две части, которые далее рекомбинируются. В двухочечном алгоритме используются две точки разрыва и, следовательно, хромосомы делятся на три части. В однородном кроссовере случайным образом генерируется строка B длиной D с двоичными значениями генов, где D -длина хромосомы. Если bi = 0 (bi$\in$B, i$\in$[1,D]), то i-е значение гена в первом потомке

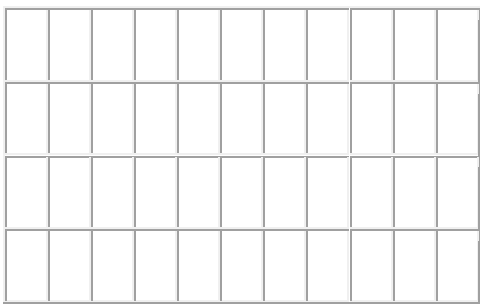
берется от первого родителя (во втором потомке — от второго родителя), если bi=1, то в первый потомок включается ген второго родителя (во второй потомок — от первого родителя).

Во-вторых, отметим ситуации, когда на состав аллелей наложены некоторые дополнительные условия. Например, пусть в задаче разбиения графа число вершин в подграфах $\mathbf{A}_1$ и $\mathbf{A}_2$ должно быть N_1 и N_2 и пусть k-й аллель, равный 1, означает, что вершина k попадает в $\mathbf{A}_1$, если же k-й аллель равен 0, то в $\mathbf{A}_2$. Очевидно, что число единиц в хромосоме должно равняться N_1, число нулей — N_2. Тогда при рекомбинации левый участок хромосомы берется от одного из родителей без изменений, а в правом участке (от другого родителя) нужно согласовать число единиц с N_1 тем или иным способом.

Один из способов — *метод PMX* (Partially Matched Crossover). Для иллюстрации PMX рассмотрим пример двухточечного кроссовера в задаче, когда в хромосоме должны присутствовать, причем только по одному разу, все значения генов из заданного набора. Пусть в примере этот набор включает числа от 1 до 9.

В табл. 1 первые две строки представляют родительские хромосомы. Третья строка содержит хромосому одного из потомков, сгенерированного в результате применения двухточечного кроссовера (после второго и пятого локусов). Полученная хромосома не относится к числу допустимых, так как в ней значения генов 1, 2 и 9 встречаются дважды, а значения 3, 4 и 5 отсутствуют. Четвертая строка показывает результат применения PMX. В этом методе выделяются сопряженные пары аллелей в одноименных локусах одной из рекомбинируемых частей. В нашем примере это пары (3 и 1), (4 и 9), (5 и 2). Хромосома потомка просматривается слева-направо; если повторно встречается некоторое значение, оно заменяется на сопряженное значение. Так, в примере в локусах 3, 5 и 9 повторно встречающиеся аллели 1, 2 и 9 последовательно заменяются на значения 3, 5 и 4.

Таблица 1



Компоновка

Содержательной сутью задачи *компоновки* может быть распределение работ по исполнителям, оборудования по помещениям, прикладного программного обеспечения и баз данных по подсетям виртуальной локальной вычислительной сети и т.п.

Рассмотрим задачу компоновки оборудования. В частности, это может быть задача размещения микросхем по модулям (платам или типовым элементам замены), модулей по панелям, панелей по шкафам радиоэлектронной аппаратуры или приборов по отсекам корабля и т.п.

Пусть задача характеризуется следующими исходными данными:

- **P** — множество элементов (конструктивов) $P_i, i = 1...n$;
- **B** — множество блоков $B_j, j = 1...m$;
- **C** — множество связей, связь C_{ik} соединяет элементы P_i и P_k.
- **D** — матрица размера $n \times n$, элемент которой D_{ik} равен числу связей между элементами P_i и P_k.

Опишем множество альтернатив **A**. Оно представляется матрицей **X** размера $m \times n$, элемент которой $X_{ji} = 1$, если конструктив P_i включен в блок B_j, иначе $X_{ji} = 0$. Некоторое распределение единиц и нулей по клеткам матрицы **X** представляет одну конкретную компоновку, т.е. одно проектное решение. Но только такие варианты распределения 1 и 0 по клеткам матрицы **X** допустимы, которые удовлетворяют следующим ограничениям:

$$\sum X_{ji} = 1, \qquad (1)$$

где суммирование производится для конструктива P_i по всем j от 1 до m;

$$\sum X_{ji} \leqslant V_{\max}, \qquad (2)$$

где суммирование производится для блока B_j по i от 1 до n, $V_{\max}$ — максимальное число элементов, помещающихся в один блок.

Первое из этих условий говорит о том, что элемент может быть помещен только в один блок. Второе условие свидетельствует об ограниченном объеме блоков.

В общем случае в задачах компоновки может быть несколько критериев. В частном случае используется единственный критерий — число межблочных связей. Число таких связей нужно минимизировать.

Для вычисления критерия применяют следующую модель.

Формируются матрицы $\mathbf{Y}=\mathbf{X}\times\mathbf{P}$ и $\mathbf{W}=\mathbf{Y}\times\mathbf{X}^{\mathrm{T}}$. Матрица $\mathbf{Y}$ размера $m\times n$ есть матрица связей блоков и конструктивов, элемент y_{ji} этой матрицы равен числу связей j-го блока с i-м конструктивом. Матрица $\mathbf{W}$ имеет размер $m\times m$, ее элемент W_{pq} равен числу связей между блоками B_p и B_q, $\mathbf{X}^{\mathrm{T}}$ — транспонированная матрица $\mathbf{X}$. Так как число межблочных (внешних) связей равно общему числу связей минус число внутренних связей K, то в качестве целевой функции, подлежащей максимизации, можно принять суммарное число внутренних связей, т.е. функцию

$$K(\mathbf{X})=\sum W_{pp}(\mathbf{X}), \qquad (3)$$

где суммирование выполняется по j от 1 до m.

Таким образом, задача компоновки представлена как задача дискретного (булева) математического программирования с целевой функцией (3), ограничениями (1), (2) и множеством управляемых параметров $\mathbf{X}$.

Аналогичным образом, формулируется ряд других задач структурного синтеза.

При решении задачи компоновки генетическим методом может быть принята хромосома следующей структуры — гены соответствуют конструктивам, значение i-го гена есть номер блока, в который помещен i-й конструктив.

В случае применения НСМ декомпозиция общей задачи приводит к локальным подзадачам, в каждой из которых выбирается один из еще не распределенных конструктивов и назначается в один из блоков. Нужно сформулировать правила S_i выбора в каждой подзадаче конструктива и правила Q_k выбора блока для этого конструктива. Каждая эвристика включает по одному правилу S_i и Q_k.

Формулировка правил — ответственная задача, от качества решения которой зависит эффективность поиска оптимума. Однако ее решение в НСМ оказывается проще, чем в традиционных эвристических методах.

Действительно, в традиционном эвристическом методе нужно сформулировать единственную комплексную эвристику, в которой с некоторыми весами были бы учтены все требования к синтезируемому решению. Но для получения решения, близкого к оптимальному, эти веса должны изменяться от шага к шагу (от подзадачи к подзадаче), а найти

корректный алгоритм изменения весов в рамках обычных эвристических методов невозможно. Веса принимаются постоянными, решение оказывается далеким от оптимума.

Метод НСМ можно рассматривать именно как генетический метод определения оптимальной последовательности весов. Но вместо комплексной эвристики проще и удобнее использовать множество простых правил и вместо изменения весов производить замену правил при переходе от шага к шагу, что и делается в НСМ. Формирование таких правил не представляет существенных трудностей, так как не нужно назначать веса. Важно лишь обеспечить разнообразие правил, чтобы рациональные решения не оказались за пределами поискового пространства.

В многокритериальных задачах оптимизации проблема формирования эвристик часто решается довольно просто: каждое правило должно соответствовать одному из имеющихся критериев оптимальности. Конечно, метод НСМ предоставляет возможности использования любых разумных правил и, следовательно, формирования набора эвристик по предпочтениям пользователя. Это обстоятельство успешно используется, когда исходные критерии оптимальности нелегко трансформировать в частные критерии локальных подзадач.

Примеры правил в задаче компоновки для выбора конструктива:

- S_1) выбирается конструктив, которому соответствует максимальный элемент в матрице $\mathbf{D}$;
- S_2) выбирается конструктив k с максимальной числом связей с другими конструктивами, т.е. с максимальной суммой элементов в k-й строке матрицы $\mathbf{D}$ (в этом и предыдущем правилах строки и столбцы уже распределенных конструктивов не учитываются);
- S_3) выбирается конструктив с максимальным числом связей с уже распределенными конструктивами, т.е. конструктив, которому соответствует столбец в матрице $\mathbf{Y}$ с максимальной суммой элементов.

Примеры правил выбора блока:

- Q_1) выбирается минимально загруженный блок;
- Q_2) выбирается максимально загруженный блок;
- Q_3) выбирается блок, имеющий максимальное число связей с распределяемым конструктивом, т.е. блок с максимальным элементом y_{ji} текущей матрицы $\mathbf{Y}$ в i-м столбце, где i — номер распределяемого конструктива.

Пару правил S_i и Q_k называют эвристикой $Э_{ik}$. В случае задачи с приведенными выше примерами правил имеем $3 \times 3 = 9$ возможных эвристик.

Кластеризация

Иногда требуется равномерное распределение элементов по имеющимися блокам. Тогда возникает задача кластеризации, которую можно рассматривать как разновидность задачи компоновки, отличающуюся типом ограничений: вместо ограничения типа неравенства на объем внутренних для кластера (блока) связей вводится ограничение типа равенства на число элементов (компонентов) в кластере. Т.е. ограничение (2) принимает вид

$$n_j = \text{ent}(n/m) \text{ или } n_j = \text{ent}(n/m) + 1; \quad (4)$$

$$\sum_{j=1}^{m} n_j = n,$$

где n_j — число элементов в j-м кластере, $\text{ent}(n/m)$ — целая часть частного от деления числа элементов n на число кластеров m. Другими словами, требуется равномерное распределение элементов по кластерам, минимизирующее связность кластеров друг с другом. В частности, в некоторых алгоритмах размещения микросхем на платах предварительно решается задача дихотомической кластеризации — разделения множества микросхем на два минимально связанных подмножества одинаковой (при четном n) мощности.

Очевидно, что в задаче кластеризации после соответствующей корректировки можно использовать правила выбора очередного элемента и его размещения в одном из кластеров, аналогичные правилам в задаче компоновки. Так, в правиле Q_1 загрузка кластера будет оцениваться не его внутренним трафиком (связностью), а числом размещенных в нем элементов, во всех правилах Q_1, Q_2 и Q_3 возможным кластером-кандидатом может быть только тот кластер, в котором еще не исчерпан лимит на размещение. Правила $S_1 - S_3$ используются без изменений. При вычислениицелевой функции штрафы за нарушение ограничений не предусматриваются, так как учет ограничений введен в эвристики.

Возможны и другие варианты множества эвристик. Например, можно использовать правила, в которых очередным размещаемым элементом выбирается элемент:

- S_1) в строке которого в матрице связей находится максимальный элемент матрицы;
- S_2) имеющий максимальную суммарную связность с уже размещенными элементами;
- S_3) имеющий минимальную суммарную связность с еще не размещенными элементами.

Для выбранного по одному из правил $S_1 - S_3$ элемента i определяется кластер, в котором:

- Q_1) имеется элемент, максимально связанный с i;
- Q_2) уже размещенные элементы максимально связаны с i (максимальна суммарная связность).

Из этих правил можно скомбинировать 6 разных эвристик и добавить эвристику, в которой сначала определяется кластер L по признаку минимальной загруженности, а затем для него выбирается элемент по признаку максимальной связности с уже размещенными в L элементами. Ограничения на максимально допустимое число элементов в кластере введены в эвристики.

Синтез расписаний

Задачи синтеза расписаний требуется решать во многих приложениях, например, при планировании различных производственных процессов, распределении ресурсов, выборе числа и типовпроцессоров для многопроцессорных специализированных комплексов и т.п.

Рассмотрим постановку и подход к решению одной из основных разновидностей задач синтеза расписаний, обозначаемой *JSSP* (Job Shop Scheduling Problem). К исходным данным относятся:

- множество работ $\mathbf{A} = \{A_1, A_2 \ldots A_n\}$;
- множество исполнителей, называемых также машинами (или серверами) $\mathbf{M} = \{M_1, M_2 \ldots M_m\}$;
- матрица $\mathbf{P} = [P_{ij}]$, где P_{ij} — затраты времени на выполнение работы i на машине j;
- вектор $\mathbf{C} = (C_1, C_2 \ldots C_m)$, где C_j — цена за единицу времени работы машины M_j;
- ограничения на время окончания работ $\mathbf{T} = (T_1, T_2 \ldots T_n)$, где T_i — предельное время для завершения работы i;

- штраф Q, добавляемый к общим затратам Z при нарушении какой-либо работой соответствующего ей временного ограничения.

Требуется составить расписание, при котором минимизируются затраты на выполнение всех работ с учетом штрафов.

В других разновидностях задач синтеза расписаний могут быть те или иные усложняющие решение отличия. Примерами отличий могут быть:

1. многостадийность — наличие нескольких стадий обслуживания каждой работы;
2. разделение работ на несколько групп и учет дополнительных затрат на переналадку машин, требующейся при последовательном обслуживании работ разных групп;
3. частичная упорядоченность работ, т.е. для некоторых пар работ i, k введены ограничения вида: $B_i \geqslant E_k$,

где B_i — время начала i-й работы, E_k — время окончания k-й работы.

Каждая эвристика при применении для синтеза расписаний метода НСМ включает два правила: одно для выбора очередной работы, другое для выбора машины, обслуживающей эту работу.

Например, при синтезе многостадийных расписаний в качестве очередной может выбираться работа:

1. с наименьшим T_i;
2. с наименьшим временем окончания обслуживания на предыдущей стадии E_i.

Примеры правил выбора машин:

1. выбирается машина, на которой обслуживание данной работы будет наиболее дешевым;
2. выбирается машина, на которой работа будет обслужена за наименьшее время.

Маршрутизация транспортных средств

Среди задач *маршрутизации транспортных средств* при перевозке различных грузов одной из наиболее трудных для решения считается *задача VRPTW* (Vehicle Routing Problem with Time Windows) — задача маршрутизации транспортных средств с временными окнами. Временным окном здесь называют временной интервал, в который нужно выполнить заказ на перевозку. В задаче заданы:

- n — число узлов-потребителей продукта (k-й узел-потребитель отождествляется с k-м заказом);

- z — число узлов-источников продукта;
- m — число серверов (транспортных средств);
- w — общее число узлов;
- V — исходное расположение потребителей, источников и серверов в узлах транспортной сети;
- $\mathbf{D}$ — матрица расстояний между узлами;
- T_{1i} и T_{2i} — нижняя и верхняя границы временного окна для выполнения i-го заказа;
- p — число типов продукта.

Кроме того, для узлов-потребителей и узлов-источников заданы объемы соответственно заказанного и имеющегося продукта каждого типа, для серверов — максимальный объем перевозимого продукта, скорость движения, цена за единицу расстояния пробега, штрафы за нарушение временных ограничений (выход за пределы временных окон), причем штрафы зависят от цены единицы продукта каждого типа и от степени нарушения ограничений.

Требуется найти график движения транспортных средств для выполнения всех заказов с минимальными затратами.

Каждая эвристика при использовании НСМ включает правила для выбора заказа (узла-потребителя), узла-источника продукта и транспортного средства (не исключено, что для выполнения конкретного заказа привлекается более чем по одному источнику и/или серверу).

Для выбора узла-потребителя можно использовать следующие правила:

- S_1) предпочтение отдается узлу с максимальным суммарным объемом заказанного продукта;
- S_2) выбирается потребитель с максимальной ценой заказанного продукта;
- S_3) выбирается потребитель с минимальной верхней границей временного окна.

В правилах выбора узла-источника продукта предпочтение может отдаваться источнику:

- Q_1) с минимальным расстоянием до выбранного в первой части узла-потребителя;
- Q_2) с минимальной суммой D расстояний до узла-потребителя и до ближайшего узла, в котором имеется свободный сервер;
- Q_3) с минимальной величиной $h = \frac{(L_1+L_2)U_0}{U}$, где L_1 и L_2 — расстояния соответственно до узла-потребителяи и до ближайшего узла, в

котором имеется свободный сервер, U_0 — суммарный объем заказанного продукта, U — суммарный объем продукта в источнике.

Эвристика дополняется правилом, в котором выбирается сервер:

- V_1) с минимальным расстоянием до узла-потребителя;
- V_2) с минимальным временем освобождения от предыдущего обслуживания;
- V_3) с минимальным временем выполнения маршрута.

Синтез топологии каналов передачи данных в вычислительных сетях

Задачи определения топологии сетей различного рода и распределения в них трафика относятся к сложным задачам структурного синтеза. Задачи, подобные рассматриваемой ниже задаче*синтеза сети каналов передачи данных*, встречаются во многих приложениях при синтезе сетей и распределения в них материальных, транспортных, информационных потоков.

Рассматривается следующая задача. Дано:

- множество $\mathbf{Y}$ узлов (вершин) сети, N — число узлов;
- трафик, для его задания используется некоторая метрика; трафик задается в виде матрицы $\mathbf{T}$, ее элемент T_{ij} есть трафик между узлами i и j;
- матрица $\mathbf{C}$ стоимостей прокладки и аренды (за прогнозируемое время эксплуатации) связи между парами узлов, C_{ij} — стоимость связи между узлами i и j;
- стоимость обслуживающего оборудования, стоимость определяется типом оборудования, а тип выбирается в зависимости от трафика в соответствующем канале.

Нужно минимизировать общие затраты на построение сети, складывающиеся из затрат на прокладку и аренду связей и на обслуживающее оборудование.

Решение задачи включает две процедуры:

1. выбор системы связей — каркаса сети;
2. перенос трафика связей, не вошедших в каркас и называемых хордами, в те или и иные связи каркаса.

В первой процедуре рассматривается полный граф $\mathbf{G}=\{\mathbf{Y},\mathbf{S}\}$, $\mathbf{S}$ — множество ребер (связей) полного графа. Каркас выбирается в виде остовного дерева, в дальнейшем к нему могут добавляться некоторые другие связи.

Алгоритм выбора дерева основан на последовательном подключении к подмножеству $\mathbf{D1}$ вершин, уже включенных в дерево, очередной вершины из

подмножества **D2** вершин, еще не включенных в дерево. Правило выбора подключаемой вершины — ее максимальная близость к дереву, т.е. в дерево включается связь (k, l) при условии

$$C_{kl} = \min_{m \in \mathbf{D2}} W_m$$

где $W_m = \min_{j \in \mathbf{D1}} C_{mj}$

Начало исполнения алгоритма — включение в **D1** двух вершин k и l, которым соответствует минимальный элемент матрицы **C**.

По окончании процедуры устраняются висячие вершины путем их соединения между собой. Для этого достаточно построить еще одно дерево, но уже на графе, включающем только висячие вершины. Таким образом, каркас будет включать связи двух построенных деревьев.

Алгоритм второй процедуры основан на классификации связей. Связи первого ранга — это связи каркаса. Связь второго ранга — это хорда (i,j) такая, что имеется вершина k и связи (i,k) и (k,j) включены в каркас. Хорда P-го ранга — это хорда (i,j) такая, что имеется вершина k и связи (i,k) и (k,j) являются связями ранга меньше P, причем одна из них имеет ранг $P-1$.

Результаты классификации хорд составляют матрицу **U**, ее элемент $U_{ij} = P_{ij}$, где P_{ij} — ранг связи (i,j).

Заполнение матрицы **U** происходит по следующему алгоритму. Сначала всем связям (i,j) первого уровня присваивается ранг $P = 1$, т.е. принимаем $U_{ij} = 1$. Далее если $U_{ij} = 0$ и имеется $k > j$ такое, что $(U_{ik} = 1 \wedge U_{jk} = P) \vee (U_{ik} = P \wedge U_{jk} = 1) = \text{true}$, то $U_{ij} := P + 1$. Алгоритм выполняется при последовательном увеличении P до полного заполнения **U**.

Перенос трафика из хорд в связи каркаса происходит по эвристикам, выбираемым генетическим методом.

Начиная со связей высшего ранга и кончая связями ранга 2, последовательно к каждой связи (i,j) применяется одна из следующих эвристик.

1. Если ($U_{ik} = 1 \wedge U_{jk} = P - 1)$, то трафик из связи (i,j) переносится в связи (i,k) и (j,k);

2. Выбирается k, соответствующее минимальной сумме $C_{ik} + C_{jk}$ при условии $(U_{ik} = P - 1 \wedge U_{jk} = 1) = \text{true}$; трафик из связи (i, j) переносится в связи (i, k) и (j, k);

3. То же, что и правило 2, но с измененным условием $(U_{ik} = P - 1 \wedge U_{jk} = P - 1) = \text{true}$;

4. То же, что и 3, но перенос трафика осуществляется только, если $C_{ij} > C_{ik} + C_{jk}$, иначе связь присоединяется к каркасу.

ТЕМА №6. Количество информации. Информационная энтропия. Коэффициент избыточности сообщения. Кодирование информации. Теоремы Шеннона.

Информация (в обыденном смысле) – это сведения об объектах или явлениях окружающей среды, которые мы запрашиваем в случае возникновения необходимости в них. В Законе Российской Федерации "Об информации, информатизации и защите информации" информация также определяется, как сведения о лицах, предметах, фактах, событиях, явлениях и процессах независимо от формы их представления.

Однако в этом определении присутствует понятие "сведения", которое тоже нуждается в определении. Поэтому появляются другие определения, например, "информацию можно охарактеризовать как формализованный продукт преобразования зарегистрированных сигналов в известные (субъекту) понятия". Кроме того, это определение подразумевает наличие запрашивающего, т.е. понятие информации является субъективным.

В настоящее время существуют две точки зрения на *информацию*:

- функциональная, в соответсвии с которой информация является отражением материального мира в сознании живых существ;
- атрибутивная, в соответсвии с которой информация является объективно существующей реальностью, такой же категорией, каковыми являются энергия и вещество.

Как отвечают на вопрос "что такое информация?" сторонники атрибутивного подхода?

Существует большое число определений информации. Одним из них является определение В.М.Глушкова: "Информация, в самом общем ее понимании, представляет собой меру неоднородности распределения материи и энергии в пространстве и времени, меру изменений, которыми сопровождаются все протекающие в мире процессы". В соответствии с этим определением понятия информации и энтропии оказываются близкими в отношении неоднородности, но В.Глушков наделяет информацию также динамическими свойствами - "мера изменений".

Количество информации в сообщении (элементе сообщения) определяется по формуле

$$I = — \log_2 P,$$

где P — вероятность появления сообщения (элемента сообщения). Из этой формулы следует, что единица измерения количества информации есть

количество информации, содержащееся в одном разряде двоичного кода при условии равной вероятности появления в нем 1 и 0. В то же время один разряд десятичного кода содержит $I = — \log_2 P = 3{,}32$ единицы информации (при том же условии равновероятности появления десятичных символов, т.е. при $P = 0{,}1$).

Информационная энтропия характеризует степень неопределенности передаваемой информации.

Информационная энтропия источника информации с N независимыми сообщениями есть среднее арифметическое количеств информации сообщений

$$H = — \sum_{k=1}^{N} P_k \log_2 P_k$$

где P_k — вероятность появления k-го сообщения. Другими словами, энтропия есть мера неопределенности ожидаемой информации.

П р и м е р. Пусть имеем два источника информации, один передает двоичный код с равновероятным появлением в нем 1 и 0, другой имеет вероятность 1, равную 2^{-10}, и вероятность 0, равную $1\text{-}2^{-10}$. Очевидно, что неопределенность в получении в очередном такте символа 1 или 0 от первого источника выше, чем от второго. Это подтверждается количественно оценкой энтропии: у первого источника H = 1, у второго приблизительно $H = -2^{-10} \log_2 2^{-10}$, т.е. значительно меньше.

Коэффициент избыточности сообщения A определяется по формуле

$$r = (I_{max} — I)/I_{max},$$

где I — количество информации в сообщении A, I_{max} — максимально возможное количество информации в сообщении той же длины, что и A.

Пример избыточности дают сообщения на естественных языках, так, у русского языка r находится в пределах 0,3...0,5.

Наличие избыточности позволяет ставить вопрос о сжатии информации без ее потери в передаваемых сообщениях.

Кодирование — представление сообщения (информации) последовательностью элементарных символов.

Рассмотрим кодирование дискретных сообщений. Символы в сообщениях могут относиться к алфавиту, включающему n букв (буква — символ сообщения). Однако число элементов кода k существенно ограничено сверху энергетическими соображениями, т.е. часто $n > k$. Так, если отношение сигнал/помеха для надежного различения уровня сигнала должно быть не менее q, то наименьшая амплитуда для представления одного из k символов должна быть qg, где g — амплитуда помехи, а наибольшая амплитуда соответственно

qgk. Мощность передатчика пропорциональна квадрату амплитуды сигнала (тока или напряжения), т.е. должна превышать величину, пропорциональную $(qgk)^2$. В связи с этим распространено двоичное кодирование с k = 2. При двоичном кодировании сообщений с n типами букв, каждая из n букв кодируется определенной комбинацией 1 и 0 (например, код ASCII).

Кодирование аналоговых сообщений после их предварительной дискретизации должно выполняться в соответствии с *теоремой Котельникова*: если в спектре функции f(t) нет частот выше F_B, то эта функция может быть полностью восстановлена по совокупности своих значений, определенных в моменты времени t_k, отстоящие друг от друга на величину $1/(2F_в)$. Для передачи аналогового сигнала производится его дискретизация с частотой отсчетов $2F_B$ и выполняется импульсно-кодовая модуляция последовательности отсчетов.

Первая теорема Шеннона декларирует возможность создания системы эффективного кодирования дискретных сообщений, у которой среднее число двоичных символов на один символ сообщения асимптотически стремится к информационной энтропии источника сообщений (при отсутствии помех).

Вторая теорема Шеннона относится к условиям надежной передачи информации по ненадежным каналам.

Пусть требуется передать последовательность символов, появляющихся с определёнными вероятностями, причём имеется некоторая вероятность того, что передаваемый символ в процессе передачи будет искажён. Теорема Шеннона. утверждает, что можно указать такое, зависящее только от рассматриваемых вероятностей положительное число v, что при сколь угодно малом e>0 существуют способы передачи со скоростью v'(v' < v), сколь угодно близкой к v, дающие возможность восстанавливать исходную последовательность с вероятностью ошибки, меньшей e. В то же время при скорости передачи v', большей v, это уже невозможно. Упомянутые способы передачи используют надлежащие "помехоустойчивые" коды. Критическая скорость v определяется из соотношения Hv = C, где H — энтропия источника на символ, C — ёмкость (пропускная способность) канала в двоичных единицах в секунду. Одной из форм представления этой теоремы может служить соотношение Хартли-Шеннона

$$C = 2F \log_2 k,$$

где C — пропускная способность (бит/с), F — полоса пропускания линии (Гц), $k \le 1 + A$, A — отношение сигнал/помеха.

ТЕМА №7. Коды для текстовых документов. Моментальные коды. Сжатие данных. Методы сжатия и форматы данных.

Широко используются двоичные коды:

EBCDIC (Extended Binary Coded Decimal Interchange Code) — символы кодируются восемью битами; популярен благодаря его использованию в IBM;

ASCII (American Standards Committee for Information Interchange) — семибитовый двоичный код.

Оба этих кода включают битовые комбинации для печатаемых символов и некоторых распространенных командных слов типа NUL, CR, ACK, NAK и др.

Для кодировки русского текста нужно вводить дополнительные битовые комбинации. Семибитовая кодировка здесь уже недостаточна. В восьмибитовой кодировке нужно под русские символы отводить двоичные комбинации, не занятые в общепринятом коде, чтобы сохранять неизменной кодировку латинских букв и других символов. Так возникли кодировка КОИ-8, затем при появлении персональных ЭВМ — альтернативная кодировка и при переходе к Windows — кодировка 1251. Множество используемых кодировок существенно усложняет проблему согласования почтовых программ в глобальных сетях.

Для разметки текстовых документов в Web-технологиях широко применяются коды HTML и XML.

Моментальный код - это код, в котором кодовые слова не зависят от последующих символов сообщения, т.е. символы сообщения могут кодироваться и декодироваться сразу после их прихода в кодек. Код является моментальным, если выполняется неравенство Крафта

$$\sum_{i=1}^{m} r^{-q_i} \leqslant 1,$$

где r - число знаков в алфавите используемого кода, m - число символов источника, q_i - длина i-го кодового слова.

Например, в случае двоичных блок-кодов, т.е. при r=2 и одинаковых q_i, равных 8, имеем $m \leqslant 64$, т.е. восьмиразрадным двоичным кодом можно закодировать 64 различных символа.

Представление сообщения в виде кода с уменьшенным числом символов за счет уменьшения избыточности или за счет потери малосущественной информации называется *сжатием* (компрессией) данных. Мерой избыточности является коэффициент избыточности

сообщения A, определяемый по формуле

$$r = \frac{I_{\max} - I}{I_{\max}},$$

где I — количество информации в сообщении A, $I_{\max}$ — максимально возможное количество информации в сообщении той же длины, что и A.

Пример избыточности дают сообщения на естественных языках, так, у русского языка r находится в пределах 0,3...0,5.

Благодаря избыточности возможно сжатие информации без ее потери в передаваемых сообщениях. Для этого используются специальные алгоритмы сжатия, уменьшающие избыточность. Эффект сжатия оценивают *коэффициентом сжатия*:

$$K = \frac{n}{q},$$

где n — число минимально необходимых символов для передачи сообщения (практически это число символов на выходе эталонного алгоритма сжатия); q — число символов в сообщении, сжатом данным алгоритмом. Часто степень сжатия оценивают отношением длин кодов на входе и выходе алгоритма сжатия.

Наряду с методами сжатия, не уменьшающими количество информации в сообщении, применяются методы сжатия, основанные на потере малосущественной информации.

Компрессия (сжатие) и декомпрессия данных осуществляются либо на прикладном уровне с помощью программ сжатия, либо с помощью аппаратных средств непосредственно в составе звуковых карт, видеобластеров, модемов и т.п. Устройства или программы, применяемые для компрессии и декомпрессии, называют *кодеками*. Слово "кодек" образовано из начальных слогов слов "кодирование-декодирование".

Очевидный способ сжатия числовой информации, представленной в коде ASCII, заключается в использовании сокращенного кода с четырьмя битами на символ вместо восьми, так как передается набор, включающий только 10 цифр, символы "точка", "запятая" и "пробел".

Одну из групп методов сжатия данных при аналого-цифровом преобразовании составляют методы *разностного кодирования*. В этих методах передаются разности чисел, представляющих соседние отсчеты (значения амплитуд). Сжатие обусловлено тем, что разности амплитуд представляются меньшим числом разрядов, чем сами амплитуды. Разностное кодирование реализовано в методах дельта-модуляции и ее разновидностях.

В случае *дельта-модуляции* передаваемые разности амплитуд представляются однобитовыми кодами, изображающими знаки "плюс" и "минус", причем "плюс" означает увеличение, а "минус" — уменьшение амплитуды по сравнению с предыдущим значением на величину, соответствующую одному уровню дискретизации. Очевидно, что частота дискретизации при дельта-модуляции должна соответствовать скорости изменения кодируемой величины, так как при недостаточной частоте потребуется кодирование более чем одним разрядом.

Предсказывающие (предиктивные) методы основаны на экстраполяции значений амплитуд отсчетов, и если выполнено условие

$$|A_r - A_p| > d,$$

то отсчет должен быть передан, иначе он является избыточным; здесь A_r и A_p — амплитуды реального и предсказанного отсчетов, d — допуск (допустимая погрешность представления амплитуд). Иллюстрация предсказывающего метода с линейной экстраполяцией представлена рис. 1. Здесь точками показаны предсказываемые значения сигнала. Если точка выходит за пределы "коридора" (допуска d), показанного пунктирными линиями, то происходит передача отсчета. На рисунке передаваемые отсчеты отмечены темными кружками в моменты времени t_1, t_2, t_4, t_7. Если передачи отсчета нет, то на приемном конце принимается экстраполированное значение.

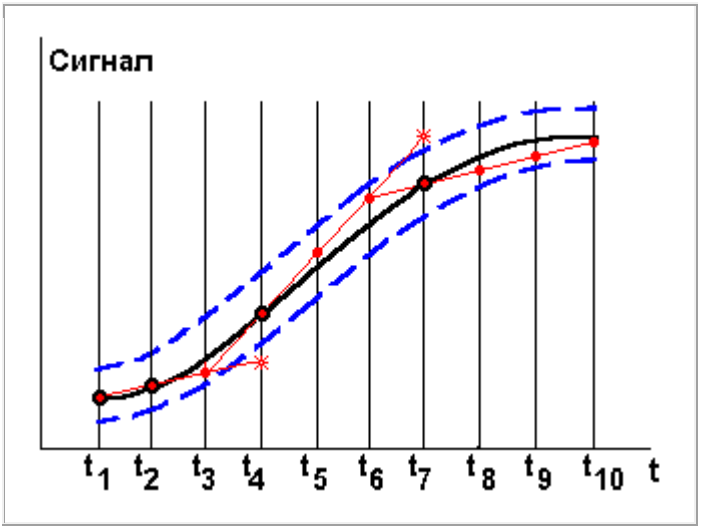


Рис. 1. Предиктивное кодирование

Для сжатия текстовой информации часто применяют *метод Хаффмена*, относящийся к статистическим методам сжатия. Идея метода — часто повторяющиеся символы нужно кодировать более короткими цепочками битов, чем цепочки редких символов. Строится двоичное дерево, листья соответствуют кодируемым символам, код символа представляется последовательностью значений ребер (эти значения в двоичном дереве суть 1 и

0), ведущих от корня к листу. Листья символов с высокой вероятностью появления находятся ближе к корню, чем листья маловероятных символов.

Распознавание кода, сжатого по методу Хаффмена, выполняется по алгоритму, аналогичному алгоритмам восходящего грамматического разбора. Например, пусть набор из восьми символов (A, B, C, D, E, F, G, H) имеет следующие правила кодирования:

$A ::= 10; B ::= 01; C ::= 111; D ::= 110; E ::= 0001; F ::= 0000; G ::= 0011; H ::= 0010.$

Тогда при распознавании входного потока 101100000110 в стек распознавателя заносится 1, но 1 не совпадает с правой частью ни одного из правил. Поэтому в стек добавляется следующий символ 0. Полученная комбинация 10 распознается и заменяется на A. В стек поступает следующий символ 1, затем 1, затем 0. Сочетание 110 совпадает с правой частью правила для D. Теперь в стеке AD, заносятся следующие символы 0000 и т.д.

Недостаток метода заключается в необходимости знать вероятности символов. Если заранее они не известны, то требуются два прохода: на одном в передатчике подсчитываются вероятности, на другом эти вероятности и сжатый поток символов передаются к приемнику. Однако двухпроходность не всегда возможна.

Этот недостаток устраняется в однопроходных алгоритмах адаптивного сжатия, в которых схема кодирования есть схема приспособления к текущим особенностям передаваемого потока символов. Поскольку схема кодирования известна как кодеру, так и декодеру, сжатое сообщение будет восстановлено на приемном конце.

Для сжатия графической и видеоинформации широко используют методы RLE (Run Length Encoding), LZW (Lempel, Zif, Welch) или LZ, JPEG (Joint Photographic Expert Group) и MPEG (Moving Pictures Experts Group).

В *методе RLE* вместо передачи цепочки из одинаковых символов передаются символ и значение длины цепочки. Метод эффективен при передаче растровых изображений, но малополезен при передаче текста. На основе этого метода созданы графические форматы TIFF, BMP, PCX.

В *LZW-методах* (алгоритмах Лемпеля-Зива-Уэлча), называемых также LZ-алгоритмами, используется следующая идея: если в тексте сообщения появляется последовательность из двух ранее уже встречавшихся символов, то эта последовательность объявляется новым символом, для нее назначается код, который при определенных условиях может быть значительно короче исходной последовательности. В дальнейшем в сжатом сообщении вместо исходной последовательности записывается назначенный код. При декодировании повторяются аналогичные действия и потому становятся известными

последовательности символов для каждого кода. Один из LZW-алгоритмов применен в протоколе V.42bis, используемом для модемной связи.

Для пояснения LZW-кодирования воспользуемся примером, приведенным Д.С.Ватолиным в работе "Алгоритмы сжатия изображений".

Пусть мы сжимаем последовательность байтов 45, 55, 55, 151, 55, 55, 55. Сначала в выходной поток заносится код очистки <256>, а в таблицу символов записываются 256 строк, соответствующих каждая одному из 256 одиночных символов. Работа продолжается следующим образом. Рассматривается первый символ 45. Строка 45 в таблице символов есть. Тогда считывается следующий символ 55 из входного потока. Строки 45, 55 в таблице нет. Поэтому в таблицу записывается строка 45, 55 (с первым свободным кодом 258) и в выходной поток записывается код <45>. Можно коротко представить архивацию так:

“45” — есть в таблице; “45, 55” — нет. Добавляем в таблицу <258>“45, 55”. В поток: <45>; “55, 55” — нет. В таблицу: <259>“55, 55”. В поток: <55>; “55, 151” — нет. В таблицу: <260>“55, 151”. В поток: <55>; “151, 55” — нет. В таблицу: <261>“151, 55”. В поток: <151>; “55, 55” — есть в таблице; “55, 55, 55” — нет. В таблицу: <262> “55, 55, 55” . В поток: <259>;

Последовательность кодов для данного примера, попадающих в выходной поток: <256>, <45>, <55>, <55>, <151>, <259>.

Существенно то, что для декомпрессии не требуется передавать таблицу символов, приемник сам создаст эту таблицу тем же способом, который использовался в передатчике.

Разновидности методов LZW нашли применение в ряде форматов, например GIF, ZIP, TIFF.

Методы JPEG, используемые в одноименном *формате JPEG*, основаны на исключении из видеоданных малосущественной информации, что не приводит к сколько-нибудь заметному ухудшению качества изображений, поскольку не различимые для глаза оттенки кодируются одинаково. В методах JPEG передаваемая последовательность пикселей делится на блоки, в каждом блоке производится преобразование Фурье, устраняются высокие частоты, передаются коэффициенты разложения для оставшихся частот, по ним в приемнике изображение восстанавливается.

Формат GIF наиболее приемлем для чертежей и рисунков, он позволяет сохранить высокое качество цветных фотографий, но за счет большого объема занимаемой памяти. Формат JPEG более экономичен при представлении фотографий с богатой палитрой цветов.

Графический формат *PNG* создан как альтернатива устаревшему формату GIF и более сложному формату TIFF.

Формат PNG имеет ряд преимуществ перед форматом GIF: неограниченное количество цветов в изображении (для GIF ограничено 256), двумерная чересстрочная прогрессивная развёртка, позволяет более высокий уровень сжатия, он открыт для использования (формат GIF принадлежит фирме Compuserve, что ограничивает возможности его использования в свободном программном обеспечении).

Форматы PNG и JPEG являются основными форматами графических файлов.

В *методах MPEG* (MPEG-1 и MPEG-2) применяют предсказывающее кодирование изображений для сжатия данных о движущихся объектах вместе со звуком. Сжатие заключается в передаче только изменившихся во времени частей изображения, при этом достигается степень сжатия в несколько десятков раз. MPEG-1 используют при пропускных способностях до 1,5 -2,0 Мбит/с, MPEG-2 — до 10 и более Мбит/с. Если в формате MPEG-1 используются фиксированные частоты смены кадров (25 Гц для размеров кадров 352×288 и 384×288 пикселов или 30 Гц для размеров 352×240 и 320×240), то в форматах MPEG-2 и MPEG-4 — переменные. Телевизионные передачи ведутся с помощью разновидностей формата .MPEG-2.

Основным преимуществом метода MPEG-4 является более высокая, чем у MPEG-2, эффективность сжатия данных.

Сжатая видеоинформация упаковывается в файлы, имеющие форматы AVI, MOV, MPG, WMV и др. Формат AVI разработан для операционных систем Windows. Формат MOV (точнее формат Quick Time, имеющий расширение MOV) создан компанией Apple, он может использоваться не только с среде ОС Macintosh, но и ОС Windows. В AVI- и MOV-файлах объединяются последовательности кадров и звук. Формат MPG, реализующий методы сжатия MPEG, отличается высокой степенью компрессии видеоинформации.

Для представления звуковых данных используют другой ряд форматов — MP3, WAVE, WMA, VQF и др. К числу наиболее популярных относится *формат MP3*, в котором компрессия основана на устранении элементов, малозначимых для качества воспроизведения звука. Формат WAVE (в наименованиях файлов используется расширение wav) используется во многих звуковых картах для представления результатов аналого-цифрового преобразования.

Среди форматов представления текстовой информации наиболее популярными являются форматы DOC, PDF, HTML, XML.

Документы в формате DOC создаются с помощью редактора MS Word, входящего в комплект офисных средств MS Office.

Отличительными особенностями *формата PDF* (Portable Document Format) являются следующие свойства:

- независимость от программно-аппаратной платформы, в которой документ воспроизводится; документ представляется в таком виде, в каком он создавался;
- возможность использования гиперссылок, встраивания интерактивных элементов, звуковых и видеофрагментов, поддержка языка JavaScript.

Форматы HTML и XML соответствуют одноименным языкам разметки, они являются основными форматами для представления документов в Web.

Для уменьшения размеров файлов в форматах DOC, PDF, HTML, XML, BMP и др. при их передаче и хранении используют средства архивации, переводящие эти файлы в форматы ZIP или RAR.

ТЕМА №8. Методы MPEG. Вейвлеты. Вейвлет-преобразование.

Вейвлеты (Wavelets)— это семейство функций, которые локальны во времени и по частоте, и в которых все функции получаются из одной посредством ее сдвигов и растяжений по оси времени. Иногда вейвлеты называют всплесками. Слово "вейвлет" в переводе с английского "wavelet" означает "маленькая волна" или "рябь". Все вейвлет-преобразования рассматривают функцию в терминах колебаний, локализованных по времени и частоте. История вейвлет-преобразований началась с работ Хаара в начале двадцатого века.

Вейвлеты характеризуются следующими свойствами.

1. Нулевое среднее значение.

2. Получаются из исходной функции семейства, так называемого анализирующего вейвлета, путем масштабного преобразования и сдвига.

3. Вейвлет-преобразование должно быть обратимо.

4. Вейвлет должен быть локализован как во временной, так и в частотной областях.

Примерами являются вейвлеты Wave, "мексиканская шляпа" и Морле, показанные слева -направо на рис. 1. Вейвлет "мексиканская шляпа" определяется как

$\Psi(t)= (1-t^2)e^{-t2/2}$

и вейвлет Морле

$\Psi(t)= e^{-t2/2}e^{j\omega t}$.

Рис. 1. Примеры вейвлетов

Вейвлет-преобразования обычно делят на дискретное вейвлет-преобразование (DWT) и непрерывное вэйвлет-преобразование (CWT). Обычно DWT используется для кодирования сигналов, а CWT для их анализа.

Вейвлет преобразования в настоящее время приняты на вооружение для огромного числа разнообразных применений, нередко заменяя обычное

преобразование Фурье во многих применениях. Все вейвлет-преобразования могут рассматриваться как разновидность временно-частотного представления и, следовательно, относятся к предмету гармонического анализа. Дискретное вейвлет преобразование может рассматриваться как разновидность фильтра конечного импульсного отклика.

Непрерывное *вейвлет-преобразование* одномерной функции времени $f(t)$ имеет вид:

$$w(a, b) = a^k \int_{-\infty}^{\infty} f(t)\Psi(\frac{t-b}{a})\, dt,$$

где $w(a, b)$ — вейвлет-образ, $\Psi(t)$ — анализирующий вейвлет, a и b — параметры преобразования, k — выбираемый пользователем показатель.

Популярны следующие анализирующие вейвлеты:

$$\Psi(t) = (1-t^2)\, e^{-t^2/2};$$

$$\Psi(t) = e^{-t^2/2}\, e^{i\omega_0 t}.$$

Первый из них называется "мексиканская шляпа", второй носит название вейвлета Морле.

Обратное вейвлет-преобразование:

$$f(t) = \frac{1}{C} \int_0^{\infty} \int_{-\infty}^{\infty} \Psi(\frac{t-b}{a})\, w(a, b) \frac{1}{a^{3+k}}\, da\, db,$$

$$C = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|}\, d\omega$$

где и должно быть $C < \infty$.

Главная положительная особенность вейвлет-преобразования заключается в том, что $w(a, b)$ содержит в себе информацию как о частотных составляющих преобразуемого сигнала, так и о привязке изменений сигнала к моментам времени. Это свойство полезно используется, например, для создания эффективных алгоритмов сжатия информации — сохранять и передавать по линиям связи можно только те значения $w(a, b)$, которые относятся к изменениям данных.

ТЕМА №9. Теории эволюции. Динамические системы. Термодинамическая энтропия. Диссипативные структуры.

Окружающий нас мир находится в состоянии постоянного движения, развития. Законы эволюции мира, как и любой составляющей его системы, являются предметом изучения многих наук и, прежде всего, термодинамики, дарвиновской теории и синергетики.

Термодинамика изучает внутренние равновесные состояния макроскопических тел. В рамках термодинамики предсказывается постепенный рост неупорядоченности, снижение разнообразия, выравнивание температуры тел и, как итог, "тепловая смерть" Вселенной. Теория эволюции Дарвина относится к живому миру и основана на принципах наследственности, изменчивости, отбора. Причем для этих теорий характерно рассмотрение квазистатических состояний систем, что недостаточно для объяснения явлений окружающего нас мира и для принятия решений по экологическим, социальным, производственным проблемам. Синергетика появилась во второй половине XX века, как обобщающая теория эволюции, рассматривающая динамику поведения систем.

Синергетика — наука о законах и процессах самоорганизации систем, как основе эволюции. При этом под самоорганизацией понимают самопроизвольное усложнение структуры системы при медленном и плавном изменении ее параметров. Процессы самоорганизации ведут к росту организованности, упорядоченности систем и противостоят термодинамическим процессам роста хаотичности.

Систему, математической моделью которой является система обыкновенных дифференциальных уравнений, называют *динамической системой*. Другое название динамической системы — детерминистическая система, поскольку она соответствует принципу детерминизма, согласно которому будущее поведение системы предсказуемо, если известно текущее ее состояние. Классический пример применения детерминизма — ньютоновская "механическая" вселенная.

Решения уравнений динамической системы можно представить, как движение некоторой точки в пространстве с размерностью, равной числу зависимых переменных системы уравнений. Траекторию движения этой точки называют *фазовой траекторией* системы. Поведение фазовой траектории в смысле устойчивости показывает, что существует несколько основных типов

траекторий, когда все решения системы в конечном счете сосредотачиваются на некотором подмножестве. Такое подмножество называется *аттрактором*. Аттрактор имеет область притяжения, т.е. такое множество начальных точек, что начавшиеся в них фазовые траектории стремятся к этому аттрактору.

Динамическая система, состояние которой полностью определяется начальными условиями и внешними воздействиями в процессе развития, является *устойчивой динамической системой*. Основными типами аттракторов при этом являются:

- устойчивые предельные точки;
- устойчивые циклы (траектория стремится к некоторой замкнутой кривой);
- торы (к поверхности которых приближается траектория).

Движение точки в таких случаях имеет апериодический, периодический или квазипериодический характер.

Наряду с устойчивыми динамическими системами существуют *хаотические динамические системы*. Это тоже детерминистические системы, но их поведение предсказуемо только теоретически, поскольку для предсказания требуется знать начальное состояние системы с абсолютной точностью, что практически недостижимо.

В фазовом пространстве хаотической системы имеется область, притягивающая к себе из окрестных областей все фазовые траектории. Эти траектории имеют сложную и запутанную структуру и представляют собой незамкнутые кривые. Эта область называется *странным аттрактором* (или перемешивающим слоем). Странные аттракторы характерны только для диссипативных систем, в их фазовом пространстве движение точки является неустойчивым, любые две траектории всегда расходятся, малое изменение начальных данных приводит к различным путям развития. Следовательно, динамика систем со странными аттракторами является хаотической.

Уникальным свойством странных аттракторов является масштабная самоповторяемость. Это означает, что увеличивая участок аттрактора, содержащий бесконечное количество кривых, можно убедиться в его подобии крупномасштабному представлению части аттрактора. Для объектов, обладающих способностью бесконечно повторять собственную структуру на мелкомасштабном уровне, существует специальное название — фракталы.

Для динамических систем, зависящих от некоторого параметра, характерно, как правило, плавное изменение характера поведения при изменении параметра. Однако для параметра может иметься некоторое

критическое (бифуркационное) значение, при переходе через которое аттрактор претерпевает качественную перестройку и, соответственно, резко меняется динамика системы, например, теряется устойчивость. Потеря устойчивости происходит, как правило, переходом от точки устойчивости к устойчивому циклу (мягкая потеря устойчивости), выход траектории с устойчивого положения (жесткая потеря устойчивости), рождение циклов с удвоенным периодом. При дальнейшем изменении параметра возможно возникновение торов и далее странных аттракторов, то есть хаотических процессов.

Термодинамика — наука, изучающая внутренние равновесные состояния макроскопических тел. Предметом термодинамики являются также законы взаимопреобразования и передачи энергии. Термодинамика описывает макроскопические параметры систем без конкретных предположений относительно их микроскопического устройства.

В основе термодинамики лежат три феноменологических закона, сформулированные на основе экспериментальных данных и принимаемые как постулаты.

Нулевое (или общее) начало термодинамики — это принцип, согласно которому изолированная термодинамическая система независимо от начального состояния в конце концов приходит к состоянию термодинамического равновесия и самостоятельно выйти из него не может.

Первое начало термодинамики представляет собой закон сохранения энергии в применении к термодинамическим системам. В наиболее простой форме его выражает соотношение:

$\Delta U = \Delta Q + A$, (1)

где U — внутренняя энергия тела, причем измерять удается только изменение внутренней энергии ΔU; ΔQ — количество теплоты; A — работа, совершенная над системой.

Внутренняя энергия тела — его полная энергия за вычетом кинетической энергии тела как целого и потенциальной энергии тела во внешнем поле сил. Это энергия движения и взаимодействия частиц, из которых состоит тело. Следовательно, внутренняя энергия складывается из кинетической энергии хаотического движения молекул, потенциальной энергии взаимодействия между ними и внутримолекулярной энергии.

Второе начало термодинамики накладывает ограничения на направление термодинамических процессов, запрещая самопроизвольную передачу тепла от более холодных тел к более горячим. Также формулируется как закон возрастания энтропии.

Термодинамическая энтропия S (или просто энтропия) в термодинамике является функцией состояния термодинамической системы; её существование постулируется вторым началом термодинамики. Понятие энтропии было впервые введено в 1865 году Р.Клаузиусом. Он определил изменение энтропии термодинамической системы при обратимом процессе как отношение изменения общего количества тепла ΔQ к величине абсолютной температуры T:

$$dS = \frac{\Delta Q}{T}$$

Другое определение энтропии

$S = k \ln Z,$

где Z — термодинамическая вероятность состояния или, другими словами, число микросостояний, соответствующих данному макросостоянию (число возможных изменений в мире атомов тела в данном состоянии), k = 1,3806504*10^{-23}Дж/К — постоянная Больцмана. Чем больше Z, тем больше хаос, т.е. энтропия - характеристика хаоса, беспорядка.

Коэффициент пропорциональности k и есть постоянная Больцмана. Это выражение, определяющее связь между микроскопическими (Z) и макроскопическими состояниями (S), выражает центральную идею статистической механики.

Клаузиус, исходя из второго начала термодинамики, пришёл к выводу, что энтропия Вселенной как замкнутой системы стремится к максимуму, и в конце концов во Вселенной закончатся все макроскопические процессы. Это состояние Вселенной получило название "тепловой смерти". Однако вывод о неизбежности "тепловой смерти Вселенной" является неправомерным, поскольку любая часть Вселенной не является сама по себе замкнутой и её приближение к состоянию теплового равновесия не является абсолютным. Термодинамическое же описание Вселенной как целого возможно лишь в рамках общей теории относительности, в которой вывод о приближении энтропии к максимуму не имеет места.

Третье начало термодинамики говорит о том, как энтропия ведет себя вблизи абсолютного нуля температур. Макс Планк сформулировал третье начало термодинамики, как условие обращения в нуль энтропии при стремлении температуры к абсолютному нулю.

Замкнутые системы, выведенные из состояния равновесия, стремятся вернуться к состоянию, характеризуемому максимумом термодинамической энтропии. В отличие от них в открытых системахвозможен отток энтропии, который может уравновесить ее рост или даже превысить его. В последнем случае возникают крупномасштабные флюктуации, а при определенных

условиях в системе начинают происходить самоорганизационные процессы, создание упорядоченных структур. Такие структуры называют *диссипативными структурами* или системами.

ТЕМА №10. Хаотические системы. Бифуркации. Фракталы. Самоорганизация. Синергетика. Теория катастроф.

Динамическая система, процессы в которой характеризуются странным аттрактором, является *хаотической системой*. Динамическая система хаотична тогда и только тогда, когда у нее существует незамкнутая фазовая траектория. В отличие от устойчивой динамической системы определить состояние хаотической системы по заданным значениям времени и начальных условий невозможно.

Приведем примеры моделей хаотических систем.

1. $dx/dt = y + mx - xz$,

$dy/dt = -x$,

$dz/dt = -gz - gq(x)x^2$,

где $q(x) = 0$ при $x \geq 0$ и $q(x) = 1$ при $x \leq 0$.

2. Система уравнений Лоренца — трехмерная система нелинейных дифференциальных уравнений первого порядка вида

$$dx_1/dt = -sx_1 + sx_2,$$
$$dx_2/dt = -x_1x_3 + rx_1 - x_2, \quad (1)$$
$$dx_3/dt = x_1x_2 - bx_3.$$

В ней s, b и r — параметры. Эта система возникла в задаче о моделировании конвективного течения жидкости, подогреваемой снизу. Такое течение описывается системой дифференциальных уравнений в частных производных. Система (1) получается из нее проектированием на специальное трехмерное подпространство.

В результате численного интегрирования системы (1) Э. Лоренц обнаружил, что при s = 10, b = 8/3 и r = 28 у этой динамической системы, с одной стороны, наблюдается хаотическое, нерегулярное поведение всех траекторий и все траектории притягиваются к аттрактору рис. 1. На рис. 2 приведена зависимость x_1 от времени.

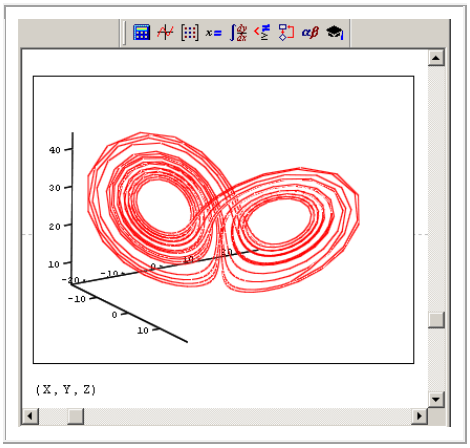


Рис. 1. Пример странного аттрактора (для задачи Лоренца)

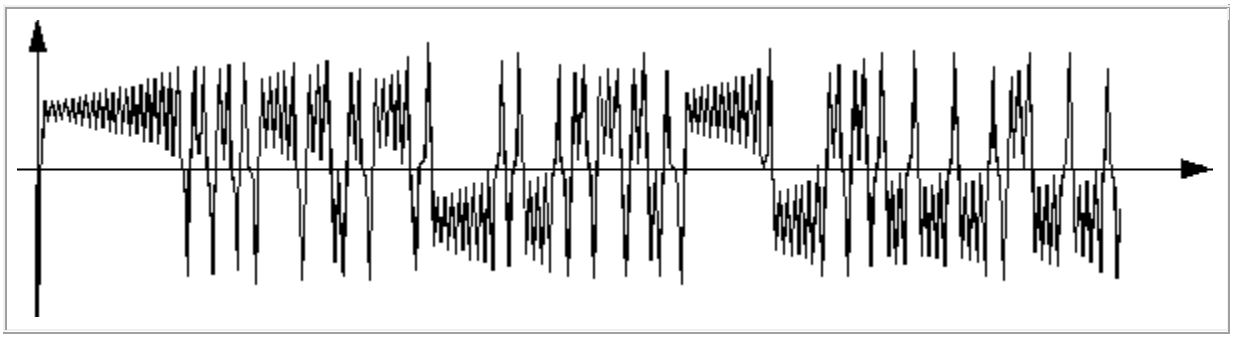


Рис. 2. Хаотические колебания (зависимость переменной x_1 от времени в задаче Лоренца)

Бифуркация - раздвоение, разделение, разветвление чего-либо. Состояние процесса в динамической системе, при котором резко возрастают флюктуации и выход из которого возможен по двум существенно различным трудно предсказуемым направлениям — хаотическому или упорядоченному.

В механических системах, как правило, установившиеся движения (положения равновесия или относительного равновесия) зависят от параметров. Значения параметров, при которых наблюдается бифуркация, называются бифуркационными значениями параметров. Кривые или поверхности, изображающие множества равновесий в пространстве состояний и параметров, называются бифуркационными кривыми или поверхностями. Прохождение параметра через бифуркационное значение, как правило, сопровождается изменением свойств устойчивости равновесий. Бифуркации равновесий могут сопровождаться рождением периодических и других, более сложных движений.

Фрактал — бесконечно самоподобная геометрическая фигура, каждый фрагмент которой повторяется при уменьшении масштаба. Другими словами, фрактал — самоподобное множество нецелой размерности. В свою очередь, самоподобное множество определяется как множество, представимое в виде объединения одинаковых непересекающихся подмножеств подобных исходному множеству. Масштабная инвариантность, наблюдаемая во фракталах, может быть либо точной, либо приближённой. Термин фрактал (лат. fractus —

дроблёный) введён Б.Мандельбротом в 1975 г. для обозначения нерегулярных самоподобных множеств.

Различают фракталы геометрические, алгебраические и стохастические.

Пример геометрического фрактала, называемого островом Коха, показан на рис. 1. Для построения алгебраических фракталов используются итерации нелинейных отображений, задаваемых простыми алгебраическими формулами, например:

$z_{i+1} = F(z_i)$, (1)

где F(z) — какая-либо функция комплексной переменной z. Фракталы, при построении которых в итеративной системе типа (1) случайным образом изменяются какие-либо параметры, называются стохастическими.

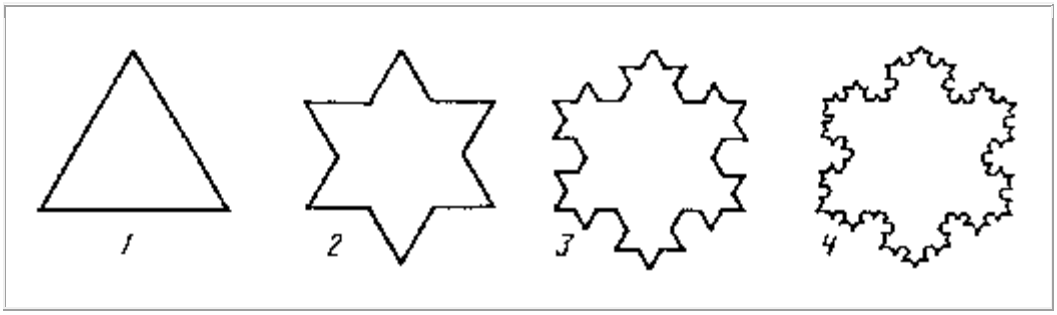


Рис. 1. Несколько первых шагов в последовательности, приводящей к построению острова Коха, который имеет ограниченную площадь и бесконечный периметр

Построение фрактала "остров Коха" можно представить следующим образом. На первом шаге берем обычный равносторонний треугольник (см. рис. 1). Потом на каждой стороне достраиваем по треугольнику, сторона которого в три, а значит, площадь в девять раз меньше, чем у исходного. И так далее. То, что получится после бесконечного количества таких шагов, называется островом Коха. Почему его побережье бесконечно? Это очень просто. На втором шаге периметр фигуры увеличится в 4/3 раза. На третьем — еще в 4/3. Это произошло потому, что каждый отрезок мы заменили ломаной, длина которой в 4/3 раза больше. А $(4/3)^n$ при n, стремящемся к бесконечности, конечно, тоже стремится к бесконечности. Если вспомнить знакомую из школьных времен геометрическую прогрессию, то можно убедиться, что площадь острова Коха конечна.

Теперь представим себе, что мы решили измерить периметр острова Коха, пользуясь линейкой определенной длины. При этом мы, конечно, будем заменять сложную изрезанную береговую линию ломаной со звеньями, не меньшими, чем наша линейка, как это всегда делают географы. Измеренный периметр будет зависеть от длины линейки. Это кажется совершенно неожиданным. Но действительно, чем меньше длина линейки, тем больше

измеренная длина побережья. Простейшая процедура измерения длины оказывается совсем не так проста, как кажется вначале.

Остров Коха обладает еще одной забавной особенностью. Допустим, что мы фотографируем этот остров в океане из космоса. Мы можем фотографировать с любым увеличением, но часть побережья будет тем меньше, чем больше увеличение. И мелкие детали в крупном масштабе, естественно, будут теряться. Типичная картина, которую мы увидим, показана на рис. 2. В крупном масштабе видим большой зубец и несколько маленьких. Увеличим маленький зубчик. То есть, по существу, увеличим маленький прямоугольничек до размеров первоначального. Опять выделим маленький прямоугольник, опять увеличим и опять увидим то же самое... И так до бесконечности. Это свойство выглядеть в любом, сколь угодно мелком масштабе примерно одинаково сейчас называется масштабной инвариантностью, а множества, которые им обладают, — фракталами. Можно спросить, как же характеризовать фракталы, если, как в сказке про Алису, размеры становятся какими-то зыбкими, ненадежными и начинают зависеть от размеров линейки?

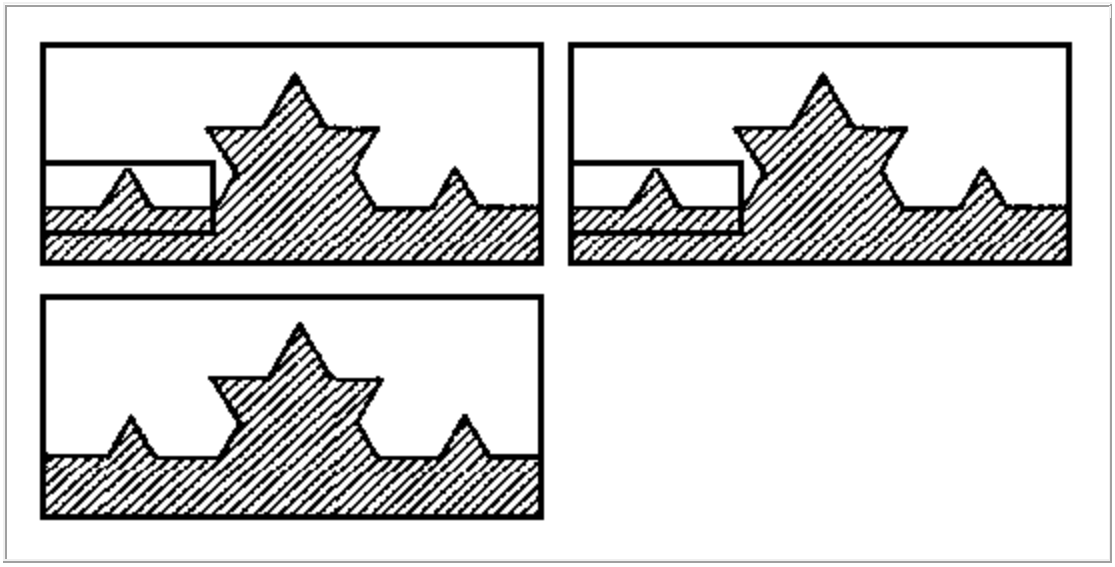


Рис. 2. Фракталы обладают масштабной инвариантностью —- при увеличении мы вновь и вновь видим одну и ту же картину. Побережье острова Коха в разных масштабах, на каждом следующем рисунке левый прямоугольник показан в увеличенном виде.

Так как фрактал состоит из бесконечного числа повторяющихся элементов, невозможно точно измерить его длину. Это означает, что чем более точным инструментом мы будем его измерять, тем большей окажется его длина. В то время как гладкая евклидова линия заполняет в точности одномерное пространство, фрактальная линия выходит за пределы одномерного пространства, вторгаясь в двумерное. Ксли для отрезка размерность равна 1, для квадрата — 2, для куба — 3, то для фракталов это —- дробное число.

Отсюда и само название "фракталы", происходящее от английского "fractal" — дробный, неполный, частичный. Например, для острова Коха оно лежит между 1 и 2. Такое значение как будто говорит, что это уже не обычная кривая, но еще не плоскость.

Самым удивительным оказывается то, что и многие природные объекты обладают как бы дробной размерностью, хотя, строго говоря, для природных объектов такую размерность вычислить невозможно. Правильнее сказать, что в определённых диапазонах наблюдения природные объекты, возникшие в результате долгой диффузии и абсорбции, похожи на фрактальные множества. Например, размерность побережья лежит между 1,01 и 1,6, а кровеносной системы человека — между 3,4 и 3,6.

Первый пример фрактала придумал классик математического анализа Вейерштрассе еще в позапрошлом веке. Так же, как к береговой линии острова Коха, к этой линии нельзя провести касательную ни в одной точке. Такие функции не имеют производной. Они вызывали у современников резкое чувство протеста. Блестящий математик Эрмит писал своему коллеге Стильтьесу: "... С омерзением и ужасом отворачиваюсь от этой зловредной язвы — непрерывных функций, нигде не имеющих производных".

И тут, наверное, рождается второе возражение: "Все это очень занятно. Но, конечно, фракталы не имеют никакого отношения к математическому моделированию реальных объектов и тем более к природе. Да и вообще математика не является естественной наукой. И ее роль не следует переоценивать". Это сильное возражение. Оно лежит в русле классической научной традиции. Следуя традиционным канонам, ценность такого математического "монстра" в познании реальности очень невелика. И хотя уже в начале прошлого века французский физик Ж.Перрен высказал мысль о том, что фракталы будут полезны во многих физических задачах, в частности, связанных с броуновским движением, к фракталам относились как к забавной математической безделице.

Ситуация кардинально изменилась с появлением в 1977 г. книги Б.Мандельброта "Форма, случай и размерность". В ней, собственно, и было введено слово "фракталы" и показано, что существование фрактальных множеств позволяет объяснить, а в некоторых случаях и предсказать экспериментальные результаты, полученные в разных областях. Среди них — космология, теория турбулентности, химическая кинетика, физика полимеров, теория просачивания жидкости и еще десятки других. В последние годы к ним прибавились физиология, физика полупроводников, теория роста городов.

Более того, даже остров Коха имеет непосредственное отношение к реальности. Английские военные топографы еще до войны заметили, что длина побережья Великобритании зависит от длины линейки, которой ее измеряют. Аналогичная зависимость определяет длину некоторых рек, побережье многих островов, путь, проходимый частицей при броуновском движении, и многое другое.

Еще пример. Оказалось, что при вытеснении жидкостью с малой вязкостью другой жидкости, с большой вязкостью, первоначально плоская поверхность раздела переходит в поверхность, напоминающую пальцы перчатки. Такие структуры получили название вязких пальцев. Последовательное дробление кончиков пальцев приводит к возникновению фрактальных кластеров. Анализ этого явления и способов борьбы с ним очень важен для приложений. Пальцы наблюдаются при закачке воды под давлением в нефтеносный пласт для повышения нефтеотдачи. Но из-за описанного эффекта вода просачивается значительно дальше, чем хотелось бы, и на поверхность выкачивается смесь, содержащая в основном воду.

Остров Коха показывает, что периметр фигуры может быть никак не связан с ее площадью. Точно так же можно построить тело с конечным объемом и бесконечной площадью поверхности. А теперь вспомним школьную химию, в которой говорится, что большинство технологических процессов требует катализа, и что в большинстве случаев он происходит на поверхности катализатора. Теперь представим себе, что нам удается создавать частицы катализатора, в определенном интервале масштабов устроенные как фракталы с бесконечной площадью. Уже появились первые сообщения о работах экспериментаторов, двигающихся по этому пути.

Этот путь от парадоксального математического объекта к обнаружению новых явлений природы в самых разных областях становится все более традиционным для неклассической науки. Именно это позволило создать новый междисциплинарный подход — теорию самоорганизации, или синергетику. В ее основе, как догадался читатель, глубокая аналогия между математическими моделями, возникающими в различных областях. Еще недавно синергетику воспринимали как моду или игру ума. Однако умение давать глубокие ответы на простые вопросы, обнаружение ряда замечательных эффектов заставили воспринимать этот подход всерьез.

Самоорганизация — процесс эволюции системы. На начальном этапе эволюции происходит медленное развитие свойств системы. Этот процесс более или менее предсказуем. В какой–то момент или внешнее воздействие достигает критического значения, или происходит кумуляция внутренних сил

(или то и другое). При этом параметры системы начинают быстро изменяться, ранее стабильное состояние резко снижает уровень стабильности, и возникает возможность разных путей развития. В этой ситуации даже незначительное воздействие может перевести эволюционный процесс на новые рельсы, развитие потом пойдёт по совсем другой линии. Наступит новый "спокойный участок", который в какой- то момент опять может смениться новым процессом бифуркации.

Бифурикационный механизм играет важнейшую роль в общей эволюционной схеме. Именно он является источником роста разнообразия различных форм организации материи, а следовательно, и непрерывно возрастающей сложности её организации. Кроме того из-за вероятностного характера бифурикационного процесса, эволюция не может иметь обратного хода, точнее, вероятность обратного хода эволюции стремится к нулю, а это имеет отношение к другому фундаментальному факту – отсутствие обратимости не только эволюции, но и времени. В этом проявляется общая направленность общего эволюционного процесса.

Синергетика — новое научное междисциплинарное направление, основанное профессором Штутгартского университета Г.Хакеном, которое занимается изучением систем, состоящих из многих подсистем различной природы и имеющих свойства, которыми не обладали подсистемы. Слово синергетика переводится как "энергия совместного действия" (от греческого: со — совместно, эргос — действие).

Синергетика представляет собой новую обобщающую науку, изучающую основные законы самоорганизации сложных систем. (математические модели явлений самоорганизации). Ее составными частями являются такие понятия и предметные области как нелинейная динамика, хаос, фракталы, катастрофы, бифуркации и т.п. Растущая в наши дни популярность синергетики объясняется тем, что она становится языком междисциплинарного общения, на котором могут друг друга понимать специалисты по математике, физике, химии, биологии, психологии.

На вопрос: "Что такое синергетика?" можно дать несколько ответов.

Во-первых, буквальный. Речь идет о явлениях, которые возникают от совместного действия нескольких разных факторов, в то время как каждый фактор в отдельности к этому явлению не приводит.

Во-вторых, синергетику часто определяют как науку о самоорганизации. Под самоорганизацией понимают самопроизвольное усложнение структуры системы при медленном и плавном изменении ее параметров. При этом

самопроизвольно возникающие образования называют диссипативными структурами.

Можно дать третье определение: синергетика — наука о неожиданных явлениях. Это определение не противоречит, а дополняет предыдущие. Действительно, при медленном плавном и монотонном изменении параметров в системе в некоторый момент "вдруг" появляются автоколебания. Причина — потеря устойчивости.

Анализ причин и законов самоорганизации и составляет предмет синергетики.

Формирование и сохранение упорядоченности структур требуется при решении многих задач не только в естественных и технических науках, но также в экономике и социологии.

Одной из практических задач синергетики является использование в искусственных системах, создаваемых человеком, явлений самоорганизации, имеющихся в биологических системах.

Вопрос об оптимальной упорядоченности и организации особенно остро стоит при исследованиях глобальных проблем — энергетических, экологических, многих других, требующих привлечения огромных ресурсов. Здесь нет возможности искать ответ методом проб и ошибок, а "навязать" системе необходимое поведение очень трудно. Гораздо разумнее действовать, опираясь на знание внутренних свойств системы, законов ее развития. В такой ситуации значение законов самоорганизации, формирования упорядоченности в физических, биологических и других системах трудно переоценить.

Основой синергетики является теория динамических систем.

В классической математической физике исследуются задачи, связанные с решением линейных уравнений. Линейные модели описывают процессы, в которых при изменении внешних воздействий наблюдаются количественные, но не качественные изменения состояний.

Если же внешние воздействия велики, то обычно начинают играть существенную роль нелинейные эффекты. Синергетика и предназначена для исследования нелинейных моделей.

Возникновение синергетики было неоднозначно воспринято научным сообществом. Одни говорили о новой парадигме в естествознании, социальных и гуманитарных науках на базе кооперации фундаментальных наук и их методов. Другие не видели в синергетике ничего нового по сравнению с современной теорией нелинейных колебаний и волн. Третьи склонялись к мнению, что синергетика всего лишь объединяющий лозунг и ничего более, и

высказывали недоумение по поводу нездорового, по их мнению, ажиотажа, вызванного новым направлением.

Столь широкий разброс мнений связан с некоторыми необычными особенностями синергетики и ее взаимосвязями с другими науками.

В отличие от наук, возникавших на стыке двух дисциплин, например, физической химии или химической физики, одна из которых предоставляет новой науке предмет, а другая — метод исследования, синергетика опирается на методы, одинаково приложимые к различным предметным областям, и изучает сложные ("многокомпонентные") системы безотносительно к их природе. Ясно, что ученый, который знакомится с синергетикой с позиции той науки, которой он занимается, прежде всего обращает внимание на те ее аспекты, которые наиболее близки основным идеям знакомой ему области знания. Что же касается отличий синергетики от наук "со стажем", то они остаются в тени. Между тем такие отличия существуют. Синергетика обращает внимание на то, что при традиционном подходе остается за рамками рассмотрения. Например, термодинамика и теория информации изучают статику, тогда как для синергетики основной интерес представляет динамика. Неравновесные фазовые переходы синергетических систем, включающие в себя колебания, пространственно-временные структуры и хаос, отличаются несравненно большим разнообразием, чем фазовые переходы систем, находящихся в состоянии теплового равновесия. В отличие от кибернетики, занимающейся разработкой алгоритмов и методов, позволяющих управлять системой так, чтобы та функционировала заданным образом, синергетика изучает самоорганизацию системы при произвольном изменении управляющих параметров. *Самоорганизацией* при этом называют процесс, идущий за счёт внутренних стимулов, не требующий вмешательства внешних факторов, не принадлежащих системе. В отличие от теории динамических систем, которая игнорирует флуктуации в точках бифуркации, синергетика занимается изучением стохастической динамики во всей ее полноте в подпространстве зависящих от времени управляющих параметров.

Важная особенность синергетических систем состоит в том, что ими можно управлять извне, изменяя действующие на системы факторы. Например, скорость роста клеток можно регулировать извне, обрабатывая клетки различными химическими веществами. Параметры, описывающие действующие на систему факторы, называются управляющими.

Временная эволюция синергетических систем зависит от причин, которые не могут быть предсказаны с абсолютной точностью. Непредсказуемость поведения синергетических систем связана не только с неполнотой

информации о состоянии их многочисленных подсистем (что заставляет ограничиваться вместо индивидуального описания каждой подсистемы описанием ансамблей подсистем) и неизбежными квантовыми флуктуациями, но и тем, что эволюция некоторых систем очень чувствительна к начальным условиям. Даже небольшое различие в начальных условиях в корне изменяет последующую эволюцию системы. Непредсказуемость эволюции синергетических систем получила название стохастичности.

В процессе временной эволюции синергетическая система, находящаяся в одном состоянии, переходит в новое состояние (старое состояние утрачивает устойчивость). При описании перехода из одного состояния в другое одни параметры состояния (быстрые переменные) можно выразить через другие (медленные переменные), которые называются параметрами порядка, в результате чего количество независимых переменных уменьшается. Возможность представления быстрых переменных в виде функций параметров порядка составляет содержание синергетического принципа подчинения. Например, если на местности имеется овраг, то самая низкая точка поверхности земли в окрестности оврага находится на его дне. Поэтому для нахождения этой точки существенны медленные переменные, или параметры порядка, описывающие "осевую" дна оврага, а быстрые переменные, описывающие склоны оврага, могут быть представлены как функции параметров порядка в силу принципа подчинения. Параметр порядка и принцип подчинения принадлежат к числу наиболее фундаментальных понятий синергетики.

Возникновение диссипативных структур носит пороговый характер. Неравновесная термодинамика связала пороговый характер с неустойчивостью, показав, что новая структура всегда является результатом раскрытия неустойчивости в результате флуктуаций. Можно сказать о "порядке через флуктуации". С математической точки зрения, неустойчивость и пороговый характер самоорганизации связаны с нелинейностью уравнений. Для линейных уравнений существует одно стационарное состояние, для нелинейных — несколько. Таким образом, пороговый характер самоорганизации связан с переходом из одного стационарного состояния в другое.

Потеря системой устойчивости называется *катастрофой*. Точнее, катастрофа — это скачкообразное изменение, возникающее при плавном изменении внешних условий. Математическая теория, анализирующая поведение нелинейных динамических систем при изменении их параметров, называется *теорией катастроф*.

Основой теории катастроф является новая область математики — теория особенностей гладких отображений, являющаяся далеким обобщением задач на

экстремум в математическом анализе. Начало было положено в 1955 г. американским математиком Г.Уитни. После работ Р.Тома (давшего теории название) началось интенсивное развитие как самой теории катастроф, так и ее многочисленных приложений. Значение элементарной теории катастроф состоит в том, что она сводит огромное многообразие ситуаций к небольшому числу стандартных схем, которые можно детально исследовать раз и навсегда.

Различают 7 канонических катастроф для функций одной или двух переменных и числа управляющих параметров, не превышающих 5.

Траектория нелинейной динамической системы в многомерном фазовом пространстве ведет себя необычным образом. При определенных условиях существует область, которая притягивает к себе все траектории из окрестных областей. Попадая в нее сколь угодно близкие траектории расходятся и имеют очень сложную и запутанную структуру. Эта область была названа "*странным аттрактором*" Лоренца. В странном аттракторе Лоренца выбранное наугад решение будет блуждать и со временем пройдет достаточно близко к любой точке аттрактора. По топологии странный аттрактор представляет собой так называемое фрактальное множество, характеризующееся дробной размерностью. Быстрое расхождение двух близких в начальный момент времени траекторий означает очень большую чувствительность решений к малому изменению начальных условий. Этим обусловлена большая трудность или даже невозможность долгосрочного прогноза поведения нелинейных динамических систем.

Теория катастроф определяет область существования различных структур, границы их устойчивости. Для изучения же динамики систем необходимо знать, каким именно образом новые решения уравнений "ответвляются" от известного решения. Ответ на такие вопросы дает теория бифуркаций (разветвлений), то есть возникновения нового решения при критическом значении параметра. Момент перехода (катастрофический скачок) зависит от свойств системы и уровня флуктуаций.

В реальных условиях при углублении неравновесности в открытой системе возникает определенная последовательность бифуркаций, сопровождающаяся сменой структур. Типичным примером такого сценария является развитие турбулентности с чередующимися типами все более усложняющихся движений. Состояние системы в момент бифуркации является неустойчивым и бесконечно малое воздействие может привести к выбору дальнейшего пути. Финальным состоянием эволюционирующих физических систем является состояние динамического хаоса.

Иллюстрацией перехода к нему является логистическое уравнение: $X_{n+1} = CX_n(1-X_n)$

Для наглядности рассмотрим биологическую трактовку этого уравнения: изолированно живет популяция особей нормированной численностью X_n. Через год появляется потомство численностью X_{n+1}. Рост популяции описывается первым членом правой части уравнения — CX_n где коэффициент с определяет скорость роста и является определяющим параметром. Убыль (за счет перенаселенности, недостатка пищи и т.п.) определяется вторым, нелинейным членом — $(CX_n)^2$.

При $C < 1$ популяция с ростом n вымирает.

В области $1 < C < 3$ численность популяции приближается к постоянному значению $X_0 = 1 - \frac{1}{C}$. Это область стационарных решений.

Затем в диапазоне $3 < C < 3.57$ появляются бифуркации, разветвление кривых на две. Численность популяции колеблется между двумя значениями, лежащими на этих ветвях. Сначала популяция резко возрастает, на следующий год возникает перенаселенность и через год численность снова становится малой. Далее происходит перекрывание областей различных решений, и поведение системы становится хаотическим. Динамические переменные X_n принимают значения, сильно зависящие от начальных. При расчетах на компьютере для близких начальных значений C решения могут резко отличаться. Более того, расчеты становятся некорректными, так как начинают зависеть от случайных процессов в самом компьютере. М.Фейгенбаум установил универсальные закономерности перехода к динамическому хаосу при удвоении периода, которые были экспериментально подтверждены для широкого класса механических, гидродинамических, химических и т.д. систем. Наряду с последовательностями удвоений периода (каскадами Фейгенбаума) имеются другие пути перехода к хаосу, когда, например, длительные периоды упорядоченного движения чередуются со вспышками беспорядка.

ТЕМА №11. Развитие систем управления предприятиями. Системы управления бизнес-процессами. Архитектурное проектирование систем. Объектно-ориентированное программирование. Компонентно-ориентированные технологии.

Начало истории АСУП относится к 60-м годам прошлого века. Все началось с планирования потребностей в материалах, представленного в системах MRP (1961 г.). Функции систем расширялись. Новые системы MRP II стали выполнять планирование в натуральных единицах и в стоимостном выражении, планирование производственных мощностей, моделирование производственных ситуаций, управление складами, снабжением, продажами, спросом и производством, что подготовило появление в 90-е годы систем ERP. В ERP-системах дополнительно реализуются финансовый учет, управленческий учет и финансовый менеджмент, бухгалтерский документооборот, бюджетирование.

По мере активного развития интернет-технологий в 90-ых появляется новое направление деятельности — электронный бизнес (e-business) — термин, которым обозначают методики и организационные принципы, позволяющие предприятию взаимодействовать со своими контрагентами через Интернет. Системы ERP с использованием Интернет-технологий стали называть *ERP-2*(Enterprise Resource and Relationship Processing — Управление ресурсами и внешними отношениями предприятия). Их функции — управление производством, продажами и финансовыми процессаи. Традиционная клиент-серверная архитектура в этих системах начинает уступать место распределенным компонентнно-ориентированным технологиям.

В настоящее время из-за расширения сфер управления и усложнения управленческих функций все труднее выполнять предъявляемые к управлению требования на базе одной системы. Это приводит к концепции открытых ERP-систем, способных интегрироваться друг с другом. Эта концепция воплощается в системах *Collaborative ERP* на основе Web и XML-технологий.

Управление бизнес-процессами (*BPM* — Buseness Performance Management или Buseness Process Management) — включает, во-первых, моделирование, анализ и планирование бизнес-процессов, во-вторых, интеграцию приложений. В BPM процессы используются и как спецификации, и как источники кода.

Для целей моделирования в BPM применяют ряд средств. К основным стандартным языкам в области бизнес-моделитрования

относятся BPEL (Business Process Execution Language), BPML(Business Process Modeling Language) и XPDL (XML Process Definition Language).

Пользователям удобно представлять разрабатываемые модели на графических языках описания бизнес-процессов, к которым относятся языки диаграмм IDEF, UML, EPC, *BPMN* (Business Process Modeling Notation).

В качестве примера диаграммного языка описания бизнес-процессов рассмотрим нотацию BPMN.

В BPMN выделяют следующие типы элементов:

- Объекты потока управления: события (рис. 1,а), действия (рис. 1,б) и логические операторы (рис. 1,в).
- Соединяющие объекты: поток управления, поток сообщений и ассоциации (рис. 1,г).
- Роли: пулы и дорожки.
- Артефакты: данные, группы и текстовые аннотации (рис. 1,д).

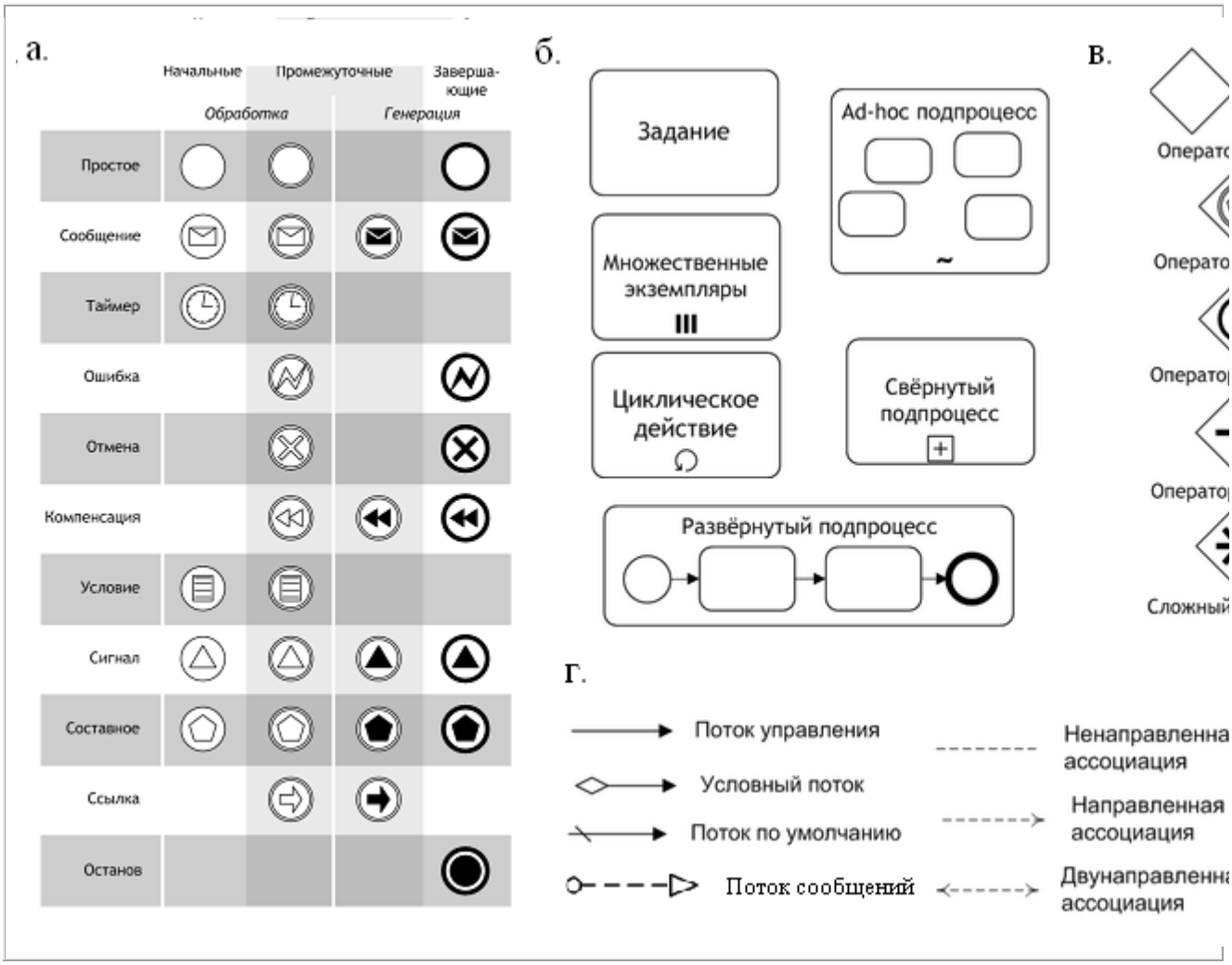


Рис. 1. Элементы диаграмм BPMN

Пример BPMN-диаграммы приведен на рис. 2.

Формальный перевод диаграмм в имитационные модели реализуется в ряде методик диаграммного моделирования. Это, например, трансляция IDEF3-моделей в цветные сети Петри, реализация UML-диаграмм в методике RUP или преобразование моделей UML в наборы BPEL и WSDL-файлов (например, с помощью программы Emerging Technologies Toolkit), переход от моделей BPMN (как выглядят такие модели видно из рис. 1) к моделям BPEL. Далее анализ процессов выполняется методами имитационного моделирования. Продвинутые пользователи часто разрабатывают модели процессов непосредственно в виде сетей Петри или на языках типа GPSS.

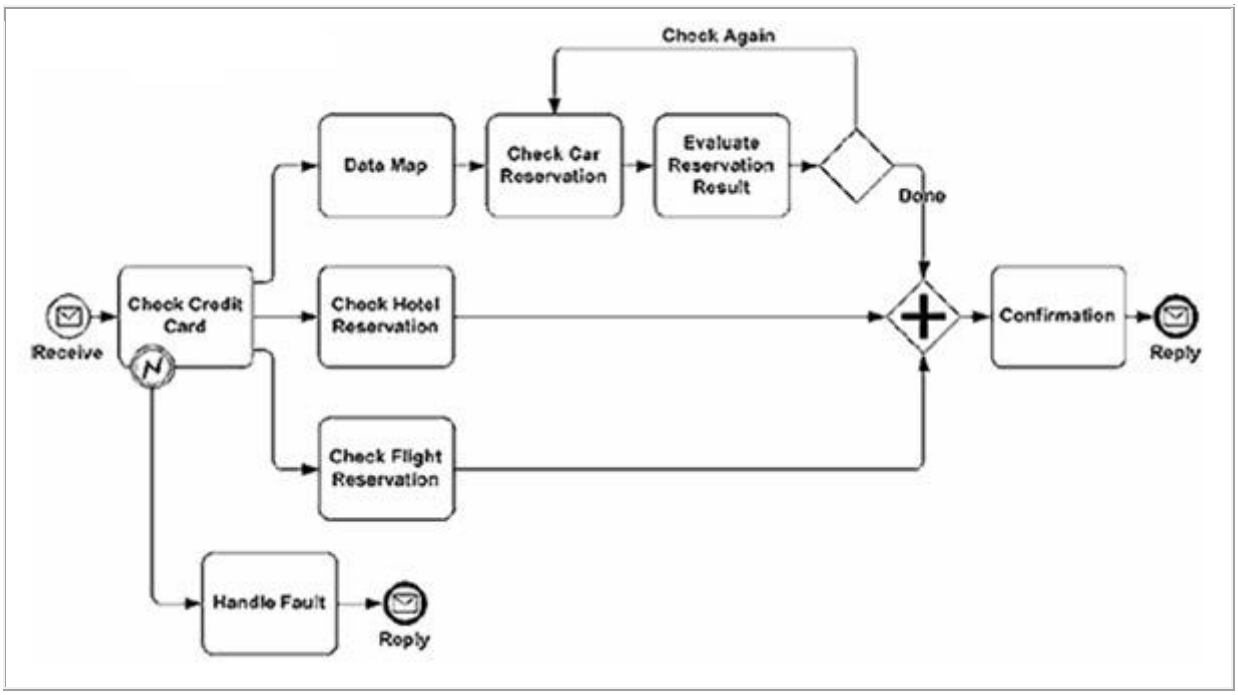


Рис. 2. Пример BPMN-диаграммы

Архитектурой называют основные принципы построения сложной системы: компьютера, компьютерной сети, автоматизированной системы, базы данных и пр. Проектирование системной архитектуры называют архитектурным или концептуальным проектированием системы.

Существует несколько парадигм архитектурного проектирования информационных систем:

1. Объектно-ориентированная (OOA, Object-Oriented Architecture);
2. Компонентно-ориентированная (CBA, Component-Based Architecture);
3. Сервисно-ориентированная (SOA. Service-Oriented Architecture) .

Объектно-ориентированный подход основан на идеях объектно-ориентированного проектирования (ООП). Структурными элементами являются классы и их экземпляры - объекты. Основными свойствами ООП являются абстрагирование, инкапсуляция, наследование и полиморфизм. Примером инструментальной среды объектно-ориентированного

проектирования программных систем может служить Rational Unified Process (RUP).

Компонентно-ориентированный подход (КОП) основан на принципах сборки программного обеспечения из независимо разработанных и повторно-используемых компонентов, т.е. структурными элементами являются компоненты. Обычно компоненты разрабатываются на основе ООП и сами представляют собой объекты, т.е. КОП является одной из форм реализации объектно-ориентированного программирования. Для КОП характерны, во-первых, распределенность компонентов в сети, во-вторых, наличие узла-посредника, который содержит сведения о интерфейсах и местоположении компонентов. Подход реализован в методиках J2EE, CORBA Component Model и Microsoft.Net и др. Особое место в СВА системах занимают многоагентные системы, компоненты которых отличаются возможностью самостоятельного автономного поведения.

Сервисно-ориентированный подход основан на использовании Web-служб, технологий и средств SOAP, WSDL., UDDI. Структурные элементы в SOA - сетевые службы, которые являются реентерабельными программами.

Следует отметить, что между подходами нет резкой границы. Так, технологию CORBA можно отнести и к ООА, а службы рассматривать как программные компоненты.

В основу *объектно-ориентированного программирования* (ООП) положены понятия класса и объекта, объединяющих в себе как данные, так и действия над ними. *Класс* - это наборсущностей, имеющих одинаковые атрибуты, операции и отношения. *Объект* - экземпляр класса. Класс предоставляет место, где общее поведение и общие переменные экземпляра могут быть определены. Переменная - это просто слот; он не содержит данных пока класс не создаст экземпляр. Класс - это фабрика для создания экземпляров. Объект существует во время выполненияпрограммы, хранит данные и операции для работы с этими данными, при этом данные объекта могут быть изменены только с помощью операций, определенных в классе.

В ООП реализуются инкапсуляция, наследованике и полиморфизм.

Инкапсуляция выполняется включением модуля в среду с помощью интерфейса — его внешнего окружения. При этом компонент рассматривается как "черный ящик": спецификации, определяющие интерфейс, выделены из модуля, а детали внутреннего содержимого скрыты от пользователя.

Наследование — это отношение, которое определяет один класс в терминах другого. В качестве примера можно привести создание подклассов-потомков на основе более общих суперклассов-предков. При этом атрибуты и

операции суперкласса автоматически присваиваются подклассу, и, кроме того, подкласс может иметь новые операции и атрибуты.

Полиморфизм — отношение, при котором различные сущности (связанные отношением наследования) по-разному реагируют на одно и то же сообщение, например, различная реакция подклассов одного класса на одно и то же сообщение. Другими словами, полиморфизм - явление, при котором один и тот же программный код выполняется по-разному в зависимости от того, объект какого класса используется при вызове данного кода. Полиморфизм обеспечивается тем, что в классе-потомке изменяют реализацию метода класса-предка с обязательным сохранением сигнатуры метода. Это обеспечивает сохранение неизменным интерфейса класса-предка и позволяет осуществить связывание имени метода в коде с разными классами — из объекта какого класса осуществляется вызов, из того класса и берётся метод с данным именем. Такой механизм называется динамическим (или поздним) связыванием — в отличие от статического (раннего) связывания, осуществляемого на этапе компиляции .

Для поддержки ООП разработаны методология и программная система RUP.

Появление *компонентно-ориентированных технологий* вызвано необходимостью повышения эффективности разработки сложных программных систем, являющихся в условиях использования корпоративных и глобальных вычислительных сетей распределенными системами. Компонентно-ориентированные технологии основаны на использовании предварительно разработанных готовых программных компонентов.

Компиляция программ из готовых компонентов — идея не новая. Уже первые шаги в области автоматизации программирования были связаны с созданием библиотек подпрограмм. Конечно, для объединения этих подпрограмм в конкретные прикладные программы требовалась ручная разработка значительной части программного кода на языках третьего поколения. Упрощение и ускорение разработки прикладного ПО достигается с помощью языков четвертого поколения (4GL), но имеющиеся системы на их основе являются специализированными и не претендуют на взаимодействие друг с другом.

Современные системы интеграции ПО построены на базе объектной методологии. Так, имеются библиотеки классов, применяя которые прикладные программисты могут создавать субклассы в соответствии с возможностями наследования, заложенными в используемые объектно-ориентированные языки программирования. При этом интероперабельность компонентов в сетевых технологиях достигается с помощью механизмов, подобных удаленному

вызову процедур RPC. К библиотекам классов относятся MFC, библиотеки для доступа к реляционным БД (например, для встраивания в прикладную программу драйверов ODBC) и др.

Преимущества использования готовых компонентов обусловлены тщательной отработкой многократно используемых компонентов, их соответствием стандартам, использованием лучших из известных методов и алгоритмов.

В то же время в компонентах библиотек классов спецификации интерфейсов не отделены от собственно кода, следовательно, использование библиотек классов не профессиональными программистами проблематично. Именно стремление устранить этот недостаток привело к появлению CBD — компонентно-ориентированных технологий разработки ПО. Составными частями таких технологий являются унифицированные способы интеграции программного обеспечения.

Возможны два способа включения компонентов (модулей) в прикладную программу — модернизация (reenginering) или инкапсуляция (encapsulation или wrapping).

Модернизация требует знания содержимого компонента, интероперабельность достигается внесением изменений собственно в сам модуль. Такой способ можно назвать способом "белого ящика". Очевидно, что модернизация не может выполняться полностью автоматически, требуется участие профессионального программиста.

Инкапсуляция выполняется включением модуля в среду с помощью интерфейса — его внешнего окружения (оболочки — wrapper). При этом компонент рассматривается как "черный ящик": спецификации, определяющие интерфейс, выделены из модуля, а детали внутреннего содержимого скрыты от пользователя. Обычно компоненты поставляются в готовом для использования виде скомпилированного двоичного кода. Обращения к модулю возможны только через его интерфейс. В спецификации интерфейса включаются необходимые для интероперабельности сведения о характеристиках модуля — модульная абстракция. В состав этих сведений могут входить описания всех входных и выходных для модуля данных (в том числе имеющихся в модуле интерактивных команд), структура командной строки для инициализации процедур, сведения о требуемых ресурсах.

Компонентно-ориентированные системы построены на основе инкапсуляции компонентов. В архитектуре этих систем можно выделить следующие части:

1. прикладная программа (клиент), создаваемая для удовлетворения возникшей текущей потребности;
2. посредник (брокер или менеджер), служащий для установления связи между взаимодействующими компонентами и для согласования их интерфейсных данных;
3. множество компонентов, состоящих каждый из программного модуля, реализующего некоторую полезную функцию, и оболочки (интерфейса). В спецификации интерфейса могут быть указаны характеристики модуля, реализуемые методы и связанные с модулем события (например, реакции на нажатие клавиш).

Собственно интерфейс представляет собой обращения к функциям модуля, называемым в CBD-технологиях методами. Эти обращения переводятся в двоичный код, что обеспечивает при их использовании независимость от языка программирования. Один и тот же модуль может реализовывать несколько разных функций, поэтому у него может быть несколько интерфейсов или методов. Каждый новый создаваемый интерфейс обеспечивает доступ к новой функции и не отменяет прежние возможно еще используемые интерфейсы.

Схематично взаимодействие компонентов можно представить следующим образом. Клиент обращается с запросом на выполнение некоторой процедуры. Запрос направляется к посреднику. В посреднике имеется предварительно сформированный каталог (реестр или репозитарий) интерфейсов процедур с указанием компонентов-исполнителей. Посредник перенаправляет запрос соответствующему исполнителю. Исполнитель может запросить параметры процедуры. После выполнения процедуры полученные результаты возвращаются клиенту.

При этом пользователь оперирует удобными для его восприятия идентификаторами компонентов и интерфейсов, а с помощью каталога эти идентификаторы переводятся в указатели (ссылки), используемые аппаратно-программными средствами и которые однозначно определяют интерфейс в распределенной сети из многих компьютеров.

В большинстве случаев реализуется синхронный режим работы, подразумевающий приостановку процесса клиента после выдачи запроса до получения ответа.

Наиболее популярными в 90-е годы были следующие CBD-технологии.

CORBA — технология, основанная на разработанных в начале 90-х г.г.спецификациях консорциума OMG, в который вошли представители ведущих компьютерных фирм. В CORBA реализуется технология распределенных вычислений на базе программ-посредников ORB.

COM (Common Object Model) — технология, развиваемая корпорацией Microsoft на базе механизма OLE. Сетевой вариант этой технологии (для систем распределенных вычислений) известен под названием *DCOM* (Distributed COM). Объекты DCOM (в частности, объекты, которые можно вставлять в HTML-документы или к которым можно обращаться из Web-браузеров) известны под названием компонентов ActiveX. В COM/DCOM, как и в CORBA , можно использовать компоненты, написанные на разных объектно-ориентированных языках программирования. Но в отличие от CORBA в COM/DCOM остается естественная для Microsoft ориентация только на операционные системы Windows. Технология ActiveX (прежнее название OLE Automation) обеспечивает интерфейс для управления объектами одного приложения из другого. В общем плане ActiveX — технология интеграции программного обеспечения фирмы Microsoft. Например, используя эту технологию, можно в среде VBA организовать доступ к объектам AutoCAD.

В технологии COM/DCOM все объекты сгруппированы в классы и каждый класс имеет свой идентификатор CLSID, а каждый интерфейс (метод) класса — свой идентификатор. Для создания объекта (экземпляра класса) клиент обращается к серверу библиотеки COM с указанием CLSID и идентификаторов всех требуемых интерфейсов. Сервер библиотеки COM находит в таблице-реестре по CLSID адрес удаленной машины, на которой размещен запрошенный компонент, и передает ей запрос клиента. На серверной стороне создается объект (копия компонента), он активируется и возвращает клиенту указатели-ссылки на требуемые интерфейсы. Теперь клиент может многократно обращаться к методам объекта, указывая в своих запросах имена интерфейса, методов и их параметров. Обычно объект исполняется на той машине, на которой размещен компонент, но можно выбрать и другую машину, указав в запросе, например, ее IP-адрес.

В настоящее время технология DCOM считается устаревшей в связи с разработкой в Microsoft среды и технологии создания программного обеспечения (в том числе распределенных приложений) Microsoft.NET.

Enterprise JavaBeans (EJB) — технология, в которой используются компоненты, написанные на языке Java. Технологию JavaBeans отличают от COM/DCOM две особенности. Во-первых, Java — единственный в JavaBeans язык программирования. Единственность языка и притом объектно-ориентированного обусловливает сравнительную легкость освоения и применения технологии JavaBeans. Во-вторых, технология JavaBeans является платформно-независимой.

Компоненты в JavaBeans являются классами Java. Для их создания, модификации и объединения в прикладные программы имеются специальные средства в составе среды J2EE.

Для корпоративных систем больших предприятий характерно использование многих программ и программных систем, относящихся к различным аппаратно-программным платформам. В такой гетерогенной среде организация распределенных вычислений обычно вызывает определенные затруднения. Кроме того, связь по технологиям RPC или CORBA происходит только по инициативе клиента. Наличие выделяемых для DCOM или CORBA отдельных портов затрудняет решение проблемы защищенности сети от несанкционированных воздействий.

Поэтому в настоящее время все более широкое распространение приобретает технология интеграции Internet-ресурсов, основанная на протоколе SOAP (Simple Object Access Protocol). Это объектная технология, в которой объектами являются Web-службы (Web Services), а для представления обращений к Web-службам используется язык XML. Web-службой называют программный компонент, предоставляющий определенные услуги по обработке информации и взаимодействующий с распределенными клиентскими приложениями через свой внешний интерфейс. Протокол SOAP обеспечивает взаимодействие распределенных систем независимо от типа объектной модели, операционной системы или языка программирования. Благодаря использованию XML, сообщения SOAP могут передаваться посредством транспортного протокола HTTP, как правило, не закрываемого сетевыми экранами.

Пример системы интеграции Internet-ресурсов — система WesSphere Application Server компании IBM. Система поддерживает технологии CORBA и ActiveX, протокол SOAP вместе с WSDL и UDDI. Интеграция основана на использовании средств Java 2 Enterprise Edition (J2EE), позволяющих различным приложениям, написанным нв Java, обмениваться информацией и совместно обрабатывать сложные транзакции.

Развитие CBD-систем возможно в направлении дальнейшего упрощения программирования и, следовательно, сокращения сроков разработки ПО, однако это происходит за счет снижения степени универсальности соответствующих инструментальных средств. Такие более специализированные средства представляют собой группу компонентов, взаимосвязанных некоторым зависящим от приложения образом, и входят в системные среды САПР.

В общем случае компоненты системной среды объединены в несколько сценариев (потоков процедур или маршрутов), в которых выделяются точки входа для вставки специфичных пользовательских фрагментов и расширений. Имеются возможности не только вставки новых фрагментов, но и замены исходных компонентов в потоках процедур на оригинальные с сохранением интерфейса. Собственно многие системы, основанные на применении языков четвертого поколения (4GL), относятся именно к таким системным средам, в которых последовательности инкапсулированных модулей образуются с помощью операторов 4GL.

ТЕМА №12. Сетевые службы. Сервис-ориентированная архитектура. Разработка, управляемая моделями. Рефакторинг. Паттерны проектирования. Метамодель.

Термином "*Сетевая служба*" или "Сетевой сервис" принято обозначать, во-первых, программное обеспечение, предоставляющее определенные услуги по обработке информации и/или доступу к ней и взаимодействующее с распределенными клиентскими приложениями через свой внешний интерфейс, во-вторых, собственно услуги и функции, выполняемые службой. *Web-службой* (Web-сервисом) называют сетевую службу, объединяющую приложения, работающие на базе Web-технологий, с использованием открытых стандартов (таких как XML, SOAP, WSDL иUDDI) в качестве Интернет-протоколов. Другими словами, Веб-службы — это программы, доступ к которым осуществляется через Веб (то есть через протокол HTTP), а обмен данными происходит в формате XML

Примечание 1

По определению консалтинговой компании Stencil Group Web-служба представляет собой свободно связанные, многократно используемые, распределенные компоненты, которые инкапсулируют ряд функций, к которым можно обращаться посредством стандартных Интернет-протоколов.

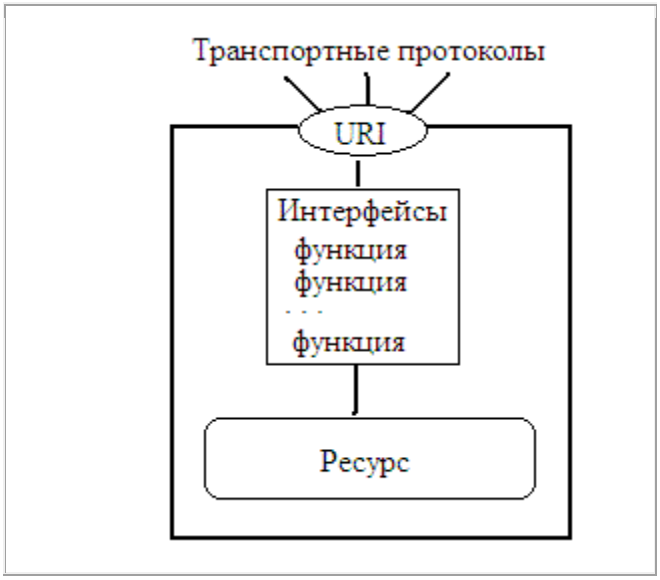


Рис. 1. Структура сетевой службы

По характеру выполняемых функций можно выделить следующие сетевые службы:

1) обеспечивающие обмен сообщениями между людьми и доступ пользователей к сетевым информационным ресурсам;

2) обеспечивающие разделение информационных ресурсов между приложениями;

3) обеспечивающие обработку информации в распределенной среде и взаимодействие сетевых приложений.

К службам первой группы относятся системы электронной почты (E-mail), средства телеконференций, аудиоконференций, видеоконференций, электронные "доски объявлений", Web-порталы, поисковые системы.

Вторая группа представлена сетевыми файловыми системами, программами поддержки спецификации ODBC, распределенными банками данных.

В третью группу входят, во-первых, системы клиент-серверной архитектуры, поддерживающие технологии активных серверных страниц, CGI, аплетов и сервлетов, во-вторых, системы с компонентно-ориентированной и сервис-ориентированной архитектурой.

Корпоративные информационные системы (КИС) могут состоять из большого числа подсистем, взаимосвязанных посредством сетевых технологий. Построение таких КИС целесообразно выполнять на основе сервис-ориентированных архитектур.

Сервис-ориентированная архитектура SOA (Service Oriented Architecture) — архитектура информационной системы, основанная на следующих принципах:

- Архитектура является распределенной, компонентно-ориентиррованной;
- Интерфейс компонентов обеспечивает динамическое обнаружение и вызов компонентов (сервисов) на основе независящего от платформы интерфейса;
- Используются общепринятые стандарты.

Согласно SOA любые компоненты информационных систем, имеющие функциональность, рассматриваются как службы (service providers, провайдеры служб), предоставляющие услуги другим частям системы.

SOA-приложения обычно представляют собой web-службы. Службы являются автономными, могут быть реализованы в разных программных средах, на разных языках программирования. Для интеграции служб в настоящее время используются XML-интерфейс, протоколы HTTP, SOAP спецификация UDDI и язык WSDL. Поиск web-служб происходит в UDDI-каталоге, интерфейс службы описан на WSDL, а вызов происходит по

протоколу SOAP. Возможны синхронные и асинхронные вызовы служб. Механизм обмена сообщениями определяется в описании сервисов на WSDL, которое которое включает форматы сообщений, типы данных, транспортные протоколы, способы сериализации, используемые при обмене между агентами заказчика и поставщика услуг. Информация о веб-сервисах содержится в реестре UDDI. Технология UDDI дает возможность поиска и публикации нужного сервиса, как человеком, так и программой-клиентом.

Сервис-ориентированная архитектура тесно связана с шиной ESB, которая является способом реализации SOA. -Однако, не все информационные системы, построенные на основе web-служб, соответствуют SOA, и SOA не обязательно должна базироваться на технологии web-служб.

Службы на основе их функциональности можно условно поделить на следующие группы:

- Службы доступа к данным, предоставляющие чтение и изменение данных. Они предоставляют единый унифицированный интерфейс для доступа к данным вне зависимости от количества и вида используемых источников данных. Для обмена данными могут использоваться специально созданные сущностные объекты, XML данные или же объекты, инкапсулирующие таблицы БД.
- Службы бизнес-логики используются для выполнения различных бизнес-операций. В ходе выполнения бизнес-операций обычно вызываются методы других служб (к примеру, служб доступа к данным для получения списка проданных товаров за последнюю неделю)
- Компоненты внешних приложений, вызываемые через их собственные интерфейсы, например, CRM система, хранящая данные о покупателях и предоставляющая доступ к этим данным черезCOM.
- Низкоуровневые вспомогательные службы, отвечающие за аутентификацию и авторизацию, мониторинг, поиск служб, регистрацию ошибок и т.п.
- составные службы, делегирующие работу остальным типам служб и координирующие их действия. Они интегрируют остальные службы системы и выполняют контроль за транзакциями и преобразование потока данных от одних служб к другим путем конвертации в нужные форматы.

Разработка, управляемая моделями, или *Model-Driven Development* (MDD) — подход к разработке, как к процессу, управляемому моделями. Другое его название — Model Driven Architecture (MDA). Управление моделями понимается в том смысле, что все основные проектные решения принимаются при оперировании моделями, а

реализация модели в виде продукта становится формальной процедурой. Этот подход развит и используется в программной инженерии.

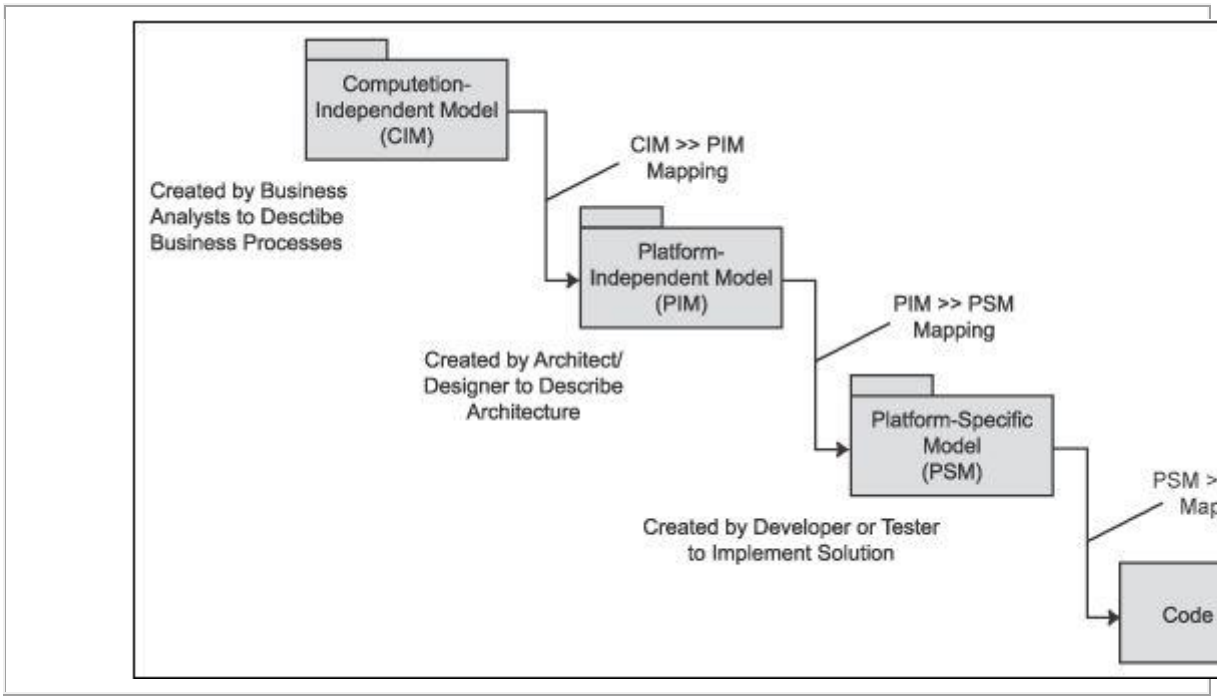
Автоматическая генерация кода на основе модели предметной области является существенным шагом на пути автоматизации разработки программного обеспечения, поскольку значительно повышает уровень абстракции, достаточный для получения работающего приложения.

Модели в MDA могут быть платформно-независимыми (PIM — platform-independent) и платформно-зависимыми (PSM — platform-specific models). Проектирование системы начинается с создания общего описания бизнес-логики приложения, т.е. модели PIM. Модель может быть выражена на высокоуровневом платформно-независимом языке программирования.

После этого осуществляется переход (трансформация) к модели PSM, в которой представлены не только функциональность системы, но и её реализация с использованием конкретной технологической платформы. Происходит дальнейшая детализация модели и добавление элементов и конструкций, специфичных для выбранной технологии реализации. В заключение выполняется генерация кода, затем производится доработка этого кода и его компиляция.

- Для поддержки MDA, в частности, создан язык UML 2.0.
- С появлением унифицированного языка моделирования UML началась активная разработка программных средств моделирования, среди которых наиболее известны Rational Rose, Together и Microsoft Visio.
- Код, генерируемый стандартными CASE-средствами, очень беден и освобождает разработчика только от механической работы по перенесению свойств модели в программу. Менее громоздкой (но более узко специализированной) является технология UniMod, позволяющая моделировать поведение системы с помощью конечных автоматов (диаграмма состояний UML), причем получаемые модели являются исполнимыми и получение готового кода также возможно. В настоящее время UniMod распространяется как модуль расширения к среде программирования Eclipse. В рамках проекта Eclipse разрабатывается библиотека EMF (Eclipse Modeling Framework), позволяющая строить метамодели и модели по ним, поддерживающая XMI и генерацию кода.

На рис. 1 показана последовательность процедур получения кода по исходному описанию бизнес-процесса [1].



- **Рис. 1.**

Рефакторинг — это изменение внутренней структуры программного обеспечения, имеющее целью облегчить понимание и упростить модификацию, но не меняющее функциональности программы.

Рефакторинг поддерживается набором методик преобразования программ. Он предназначен для решения двух основных задач: облегчение процесса повторного использования каких-либо компонентов программной системы и снижение расходов на поддержку и сопровождение системы.

Паттерн проектирования (иначе *шаблон проектирования*) представляет собой формализованное описание часто встречающейся задачи проектирования, удачное решение данной задачи, а также рекомендации по применению этого решения в различных ситуациях. Другими словами, это способ представления в общем виде как условий проектной задачи, так и правильных подходов к её решению. Паттерны предназначены для многократного использования. Важным начальным этапом при работе с паттернами является адекватное моделирование рассматриваемой предметной области. Это является необходимым как для получения должным образом формализованной постановки задачи, так и для выбора подходящих паттернов проектирования. Модель системы, построенная в терминах паттернов проектирования, фактически является структурированным выделением тех элементов и связей, которые значимы при решении поставленной задачи.

Сложные иерархированные структуры представляются как набор определенным образом типологизированных элементов и связей между ними. Кроме того, эффективной процедурой является многоуровневое представление

структур. Переход с одного уровня представления на другой осуществляется путем выделения определенных подструктур, которые, в свою очередь рассматриваются в качестве "макроскопических" элементов, связанных между собой более простым и понятным образом. В свою очередь, элементы более низкого уровня могут быть названы "микроскопическими".

Низшим уровнем представления данной системы является описание ее в терминах классов (со своими атрибутами и операциями) и соответствующих им объектов, выступающих в качестве "микроскопических" элементов, и отношений между ними, играющих роль связей. Примером "макроскопического" элемента следующего уровня является системная архитектура, представляющая собой базовую подструктуру рассматриваемой системы. Самым высоким уровнем является интеграция отдельных систем, которые в данном случае рассматриваются в качестве макроскопических элементов.

Для управляемого применения шаблона необходимо сформулировать задачу и еѐ контекст. В шаблон также может быть включен перечень условий, при выполнении которых его следует применять. Метод решения — некоторый набор элементов (классов, объектов и других объектно-ориентированных средств), с помощью которых можно задачу решить.

Типичный вид шаблона:

Имя (Контекст, Задача, Метод решения, Результаты).

Разработаны специальные библиотеки шаблонов. Более подробное описание типичной структуры шаблона дано в таблице 1.

Таблица 1

Имя шаблона	Имя в классификации
Задача	Обобщѐнная постановка задачи
Также известен как …	Другие хорошо известные имена шаблона …
Мотивация	Сценарий, иллюстрирующий проблему
Применимость	Ситуации, в которых шаблон применим
Структура	Графическая презентация шаблона
Составляющие	Классы, объекты и отношения

Сотрудничество	Распределение ответственности
Последствия	К чему приведèт применение шаблона
Реализация	Как реализовать
Образец кода	Фрагменты кода
Известные реализации	Использовано в "системе"
Родственные шаблоны	Имена родственных шаблонов

Библиотека шаблонов является справочником, который полезен как начинающим разработчикам в качестве вводного пособия, так и опытным проектировщикам как каталог типовых решений задач, часто встречающихся при разработке систем.

Метамодель в первом приближении можно определить, как модель модели. В программной инженерии более строго метамодель определяется, как некоторая целостная концепция организациидданных, определяющая структуру, интерфейс просмотра, поиска, ввода и редактирования информации. При таком определении примерами метамоделей могут служить плоские файлы, реляционные БД, электронные таблицы, многомерные кубы OLAP.

ТЕМА №13. Интегрированные среды разработки приложений. Способы интеграции информационных систем. WorkFlow. Технология SOAP. Стандарт UDDI. Язык WSDL. Средства интеграции MCAD и ERP. BizTalk Server.

CASE-системы часто отождествляют с инструментальными средами разработки ПО, называемыми также интегрированными средами разработки программного обеспечения (*IDE* — Integrated development environment) или *средами быстрой разработки приложений* (RAD — Rapid Application Development). Обычно среда разработки включает в себя текстовый и графический редакторы,компилятор и/или интерпретатор, средства автоматизации сборки, отладки, документирования программ и управления версиями. Частный случай IDE — среды визуальной разработки, которые включают в себя возможность визуального редактирования интерфейса программы.

К числу IDE относят фабрики приложений, такие как Eclipse или Microsoft Visual Studio. Примеры других сред разработки — Sun Studio, Turbo Pascal, Borland C++, Borland Delphi, VB (Visual Basic), PowerBuilder. Применение инструментальных сред существенно сокращает объем ручной работы программистов, особенно при проектировании интерактивных частей программ.

Простейшие варианты инструментальных сред представлены наборами средств разработки программ, называемыми *SDK* (Software Development Kit). Обычно SDK распространяются бесплатно с целью расширения применения определенных технологий или платформ. Пример SDK - среда разработки драйверов устройств. В случае использования языка Java SDK называют JDK (Java Developer's Kit). В JDK имеются:

1. библиотеки классов, в том числе библиотеки основных элементов языка, часто используемых оболочек (wrapper), процедур ввода-вывода, компонентов оконного интерфейса и др.
2. инструментальные средства такие, как компилятор байт-кодов, интерпретатор, просмотрщик аплетов, отладчик, формирователь оконных форм и т.п.

В средах быстрой разработки приложений RAD обычно реализуется способ программирования, называемый управлением событиями. При этом достигается автоматическое создание каркасов программ, существенно сокращается объем ручного кодирования. В этих средах пользователь может

работать одновременно с несколькими экранами (окнами). Типичными являются окна из следующего списка:

- окно меню с пунктами "file", "edit", "window" и т.п., реализующими функции, очевидные из названия пунктов;
- окно формы, на котором собственно и создается прототип экрана будущей прикладной программы;
- палитра инструментов — набор изображений объектов пользовательского интерфейса, из которых можно компоновать содержимое окна формы;
- окно свойств и событий, с помощью которого ставятся в соответствие друг другу объекты окна формы, события и обработчики событий. Событием в прикладной программе является нажатие клавиши или установка курсора мыши в объект формы. Каждому событию должна соответствовать событийная процедура (обработчик события), которая проверяет код клавиши и вызывает нужную реакцию. В RAD имеются средства для удобства разработки обработчиков событий;
- окно редактора кода, в котором пользователь записывает создаваемую вручную часть кода;
- окно проекта — список модулей и форм в создаваемой программе.

Для написания событийных процедур в Visual Basic используется язык и текстовый редактор одноименного языка, в Delphi — язык и редактор языка Object Pascal. Нужно заметить, что для реализации вычислительных процедур и, в частности, для написания миниспецификаций используется обычная для 3GL технология программирования.

Помимо упрощения написания пользовательского интерфейса, в средах RAD предусматриваются средства для реализации и ряда других функций. Так, в наиболее развитой версии Visual Basic к ним относятся средства выполнения следующих функций:

- поддержка ODBC, что дает возможность работы с различными СУБД;
- разработка баз данных;
- разработка трехзвенных систем распределенных вычислений;
- интерактивная отладка процедур на SQL Server;
- управление версиями при групповой разработке ПО;
- моделирование и анализ сценариев распределенных вычислений и др.

Платформенная инвариантность в Java достигается, благодаря введению виртуальной метамашины с системой команд, максимально приближенной к

особенностям большинства машинных языков. Любой Web-сервер при наличии запроса на Java-программу со стороны клиента транслирует (компилирует) эту программу на язык метамашины. Скомпилированный модуль, называемый*байт-кодом*, пересылается клиенту. Клиент должен выполнить интерпретацию байт-кода. Соответствующие интерпретаторы в настоящее время имеются в браузерах всех основных разработчиковWeb-технологий.

Хотя и ранее были известны технологии на базе промежуточных p-кодов, именно технология Java, оказалась наилучшим образом приспособленной для использования в гетерогенной сетевой среде. Она последовательно отражает принципы объектно-ориентированного программирования и обеспечивает приемлемую эффективность (производительность) исполнения программ. Эту эффективность можно еще более повысить, если в браузерах заменить интерпретацию на компиляцию.

Интегрированной средой разработки ПО на языке Java является J2EE.

Большинство корпоративных информационных систем в процессе своей эксплуатации нуждаются в тех или иных структурных изменениях, диктуемых необходимостью адаптации к изменяющимся условиям и требованиям бизнеса. Зачастую новые требования могут быть удовлетворены при дополнении или замене в бизнес-процессах некоторых процедур другими. Для выполнения новых процедур могут быть приобретены или использованы уже имеющиеся на предприятии прикладные программы, однако они, как правило, предназначены для автономного применения или для исполнения в иной программно-аппартной среде. Для их совместного функционирования необходимы разработка и применение соответствующих интеграционных технологий.

Способы интеграции различают по нескольким признакам.

По степени обособленности взаимосвязей подсистем в интегрируемой системе (иначе по структуре интеграции) различают варианты "точка — точка" и "звезда" (интегрирующая среда).

В варианте "точка-точка" взаимодействие подсистем осуществляется по схеме полного графа, т.е. для каждой пары взаимодействующих подсистем создается специфическая для них интерфейсная связь, например, в виде конверторов данных с языка одной подсистемы на язык другой (такую схему применительно к сетям обычно обозначают *peer-to-peer* или p2p). Поскольку число таких дуплексных связей может доходить до N(N-1)/2, где N — число подсистем, то вариант "точка-точка" оказывается приемлемым только для малых N. Подключение к системе каждой новой подсистемы оказывается весьма трудоемким.

В варианте "звезда" (рис. 1) число связей уменьшается до N, интеграция осуществляется на основе общего для подсистем языка и/или программного обеспечения, являющегося промежуточной интегрирующей средой. В отличие от способа "точка-точка" теперь достаточно в каждой подсистеме иметь интерфейс только с метасредой. Интегрирующая среда характеризуется наличием центрального компонента, управляющего взаимодействием подсистем в рамках информационной системы в целом. Интегрирующая среда выполняет сценарии транзакций, функции взаимодействия приложений, протоколирование и мониторинг состояния, обмен сообщениями между подсистемами, алгоритмы маршрутизации и доставки сообщений и др.

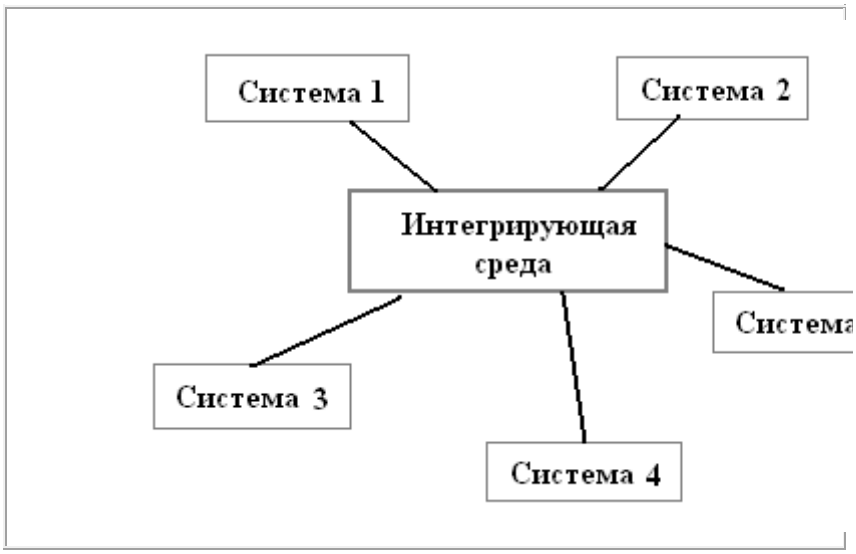


Рис. 1.

В зависимости от объема и глубины перестройки интегрируемой информационной системы различают интеграцию инфраструктуры, данных, приложений и процессов.

Интеграция инфраструктуры основана на изменениях и унификации базовых инфраструктурных элементов информационной системы — аппаратной платформы, операционной системы, службы каталогов, сетевых средств а и т.п. Такая интеграция требует больших затрат и фактически приводит к получению новой трудномодифицируемой системы.

Широкое применение имеет интеграция данных. Она не затрагивает основных инфраструктурных аспектов, за исключением средств хранения данных, основными целями подобной интеграции является обеспечение синтаксического, а иногда и семантического единства данных, организация поиска и доступа к данным и т.д. Этот способ сравнительно легко реализуем, однако избежать серьезных изменений в используемых приложениях не удается.

Интеграция данных характерна для традиционных систем клиент-сервер. При этом основным системообразующим фактором при построении информационной системы является единая база данных коллективного доступа. Интегрирующей средой является СУБД со стандартным интерфейсом доступа к данным. Все функции прикладной обработки размещаются в клиентских программах. Связь через единую БД имеет место и в системах с трехзвенной архитектурой, в которых имеется несколько прикладных подсистем и сервер баз данных.

Однако интеграция данных также имеет определенные ограничения по сложности интегрируемой системы и возможностям ее развития. Сложности модификаций снижаются в случаях применения распределенных БД, но появляется проблема синхронизации данных.

Иногда от интеграции данных отличают интеграцию информации *EII* (Enterprise information integration) в корпоративных системах, осуществляемую из многочисленных систем в унифицированное, согласованное представление и используемую для изучения и обработки данных, необходимых для отчетности и принятия решений. EII является технологией извлечения ("вытягивания") информации (pull) в отличие от проталкивания (push) при интеграции данных.

Интеграция приложений (EAI — Enterprise Application Integration) основана на использовании сервисов — общеупотребительных прикладных и системных функций, реализованных в виде серверных программ со стандартным API. В виде сервисов реализуются разнообразные функции прикладной обработки, контроля безопасности данных, файлового доступа и т.п. Интеграция приложений характеризуется не только наличием конвертации языков, но и более сложным управлением потоками данных.

Интеграция приложений возможна путем удаленного вызова процедур (рис. 2) или обмена сообщениями.

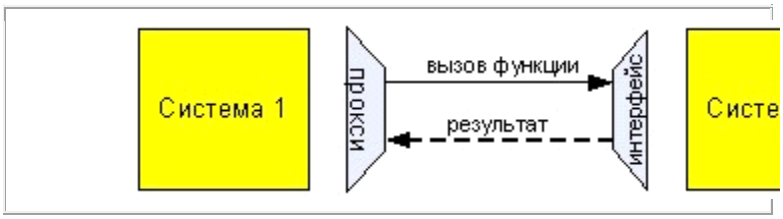


Рис. 2.

Как и в случае интеграции данных, в технологиях EAI можно выделить варианты взаимодействия по схеме p2p и интеграционной шины.

Для варианта p2p на базе языка XML разработан специальный язык обмена сообщениями WS-CDL (Web Services Choreography Description Language). С его помощью формируются запросы и ответы

взаимодействующих приложений. Поскольку вариант p2p является структурой "точка-точка", то он удобен только при малом числе связываемых приложений

Интеграционная шина (называемая также интеграционным сервером) представлет собой общую для связываемых компонентов среду передачи и обработки сообщений (рис. 1). С помощью шины осуществляется асинхронный обмен сообщениями и реализуется интеграция независимых приложений без или с минимальными доработками существующих систем. В отличие от варианта p2p, где за логику взаимодействия отвечала одна из интегрируемых систем, в данном варианте интеграцию обеспечивает интеграционная шина. Такой подход позволяет легче интегрировать новые системы, а также изменять логику интеграции, приводя ее в соответствие с бизнес-логикой процесса.

Примерами интегрирующей среды являются монитор транзакций, брокер ORB в технологии CORBA или другое программное обеспечение промежуточного слоя. Архитектура системы обычно является трехзвенной, n-звенной или компонентно-ориентированной. В интеграционном сервере возможна та или иная обработка данных, например, определение подсистемы-партнера может быть результатом анализа содержательной части сообщений.

Развитие интеграции приложений привело к появлению сервис-ориентированной архитектуры систем. Сервис-ориентированная архитектура SOA (Service Oriented Architecture) – это архитектура программной системы, обеспечивающей обнаружение и вызов сервисов с помощью не зависящего от платформы интерфейса.

Технология SOA, основанная на протоколе SOAP (Simple Object Access Protocol), — объектная технология, в которой объектами являются Web-службы (Web Services), а для представления обращений к Web-службам используется язык разметки XML. Термином "Сетевой сервис" или "Сетевая служба" в общем случае принято обозначать, во-первых, программное обеспечение, предоставляющее определенные услуги по обработке информации и/или доступу к ней и взаимодействующее с распределенными клиентскими приложениями через свой внешний интерфейс, во-вторых, собственно услуги и функции, выполняемые службой. Web-службой (Web-сервисом) называют сетевую службу (программный компонент), предоставляющую определенные услуги по обработке информации, объединяющую приложения, работающие на базе Web-технологий, использующую базовые средства Web, такие как XML и HTTP, и открытые стандарты (SOAP, WSDL иUDDI) в качестве Интернет-протоколов и взаимодействующую с распределенными клиентскими приложениями через свой внешний интерфейс

Интеграция процессов основана на организации сквозных процессов, в которых на отдельных этапах процесса задействованы те или иные приложения. Здесь не подразумевается существенное изменение отдельных приложений. При этом обработка данных на отдельных этапах может производиться в различных приложениях, а функции организации процесса и связи различных подсистем реализует специализированная подсистема. Для реализации этого способа интеграции приложений обычно применяют технологии WorkFlow.

Программное обеспечение интеграции часто объединяют под названием программ промежуточного слоя. Ведущие компании, занимающиеся корпоративными информационными системами, разрабатывают комплексные интегрирующие пакеты. К ним, в частности, относятся IBM WebSphere, Microsoft BizTalk Server, Oracle 10g, SAP Netweaver. Так, в BizTalk Server реализованы функции подготовки и транспортировки документов, документооборота.

К лингвистическому обеспечению интеграции автоматизированных систем относятся инвариантные к приложениям языки. Наиболее известными являются Express (из стандартов STEP), форматEDIFACT и язык разметки XML. Для описания бизнес-процессов разработан язык BPEL, основанный на XML. Для поддержки семантической согласованности данных используют язык описанияметаданных RDFS и языки представления онтологий приложений, такие как OWL, также основанные на XML.

Для унификации способов обмена используют транспортные протоколы. В Web-технологиях общепризнанным является протокол HTTP. С его помощью описываются запросы и ответы при клиент-серверном взаимодействии. Для определения нужного сервера и организации связи в компонентно-ориентированной среде используют протоколы SOAP, UDDI и WSDL. При этом HTTP применяют в качестве транспортного средства для сообщений SOAP.

Технологии *Workflow* (Workflow - поток работ) широко используются для интеграции процессов в информационных системах. В эти технологии входят:

- Мониторинг и обработка событий в прикладный системе.
- Обмен данными между прикладной системой и подсистемой управления процессами.
- Маршрутизация объекта вне прикладной системы
- Специализированная обработка прикладного объекта в рамках подсистемы управления процессами

- Совместное использование справочной информации

Для инициализации процесса обработки информации, порождаемой (модифицированной) в том или ином приложении, необходимы средства наблюдения за появлением или изменением состояния объекта прикладной системы в соответствии с определенными критериями и формирование события в подсистеме управления процессами.

Очень часто возникает необходимость маршрутизации того или иного объекта прикладной системы (документа, электронной формы, файла, отчета, записи справочника и пр.) При этом, доступ к данному объекту должен осуществляться не в рамках специализированного АРМа, а из общей очереди заданий подсистемы управления процессами. Например, из почтового ящика Microsoft Outlook. Примером подобной задачи может быть внесение согласовательной визирующей подписи на договор, созданный в специализированной системе согласующими лицами. Для обеспечения данной возможности объект прикладной системы должен обладать возможностью быть переданным системой маршрутизации (например, посредством электронной почты) на рабочее место пользователя. Можно выделить два типа маршрутизации On-line, при котором сам объект физически не перемещается, а маршрутизируется ссылка на объект и Off-line, при котором объект изымается из системы и физически перемещается на клиентское рабочее место для обработки.

Паттерны интеграции информационных систем относятся к верхнему уровню в классификации паттернов проектирования. Группа структурных паттернов интеграции служит для описания основных компонентов единой интегрированной метасистемы. Для описания взаимодействия отдельных корпоративных систем, включенных в интегрированную метасистему, предназначена группа паттернов, объединенных в соответствии с тем или иным методом интеграции. Интеграция корпоративных информационных систем подразумевает тем или иным способом организованный обмен данными между системами.

В настоящее время все более широкое распространение приобретает технология интеграции Internet-ресурсов, основанная на *протоколе SOAP* (Simple Object Access Protocol). SOAP —транспортный протокол, предназначенный для организации взаимодействия удаленных систем при помощи асинхронного обмена XML-документами. Первоначально SOAP предназначался, в основном, для реализации удалённого вызова процедур (RPC).

SOAP — объектная технология, в которой объектами являются Web-службы (Web Services), а для представления обращений к Web-службам используется язык XML. Язык разметки XML распознается разными системами, и потому технология SOAP значительно проще реализуется, чем такие технологии как CORBA или DCOM.

Протокол SOAP обеспечивает взаимодействие распределенных систем независимо от типа объектной модели, операционной системы или языка программирования. Благодаря использованию XML, сообщения SOAP могут передаваться посредством транспортного протокола HTTP, как правило, не закрываемого сетевыми экранами.

В стандарте, описывающем протокол SOAP, изложены принципы, по которым может быть осуществлена привязка SOAP-сообщений к абстрактному транспортному протоколу, общая схема создания SOAP-оболочек для RPC-ориентированных интерфейсов, способы возврата сообщений о сбоях, а также конкретная реализация способа оформления сообщений SOAP в качестве содержимого команд GET и POST протокола HTTP.

Структура SOAP-сообщения показана на рис. 1.

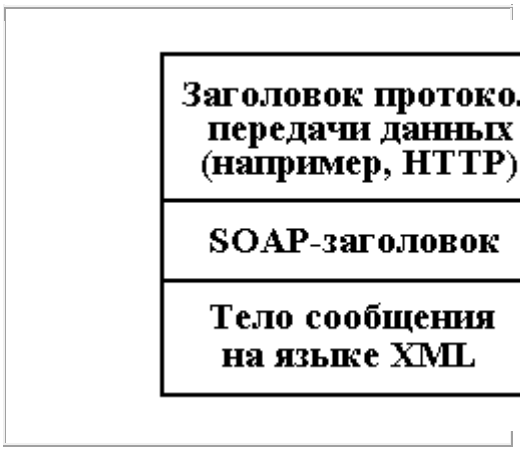


Рис. 1. Структура SOAP-сообщения

Заголовок HTTP может иметь вид:

POST/OrderEntry HTTP/1.1
Host: www.xmlbus.com
Content-Type: text/xml; charset="utf-8"
Content-Length: ...
SOAPAction: ...

В строке SOAPAction указывается URI получателя запроса или посредника.

SOAP-заголовок необязателен (но заголовков может быть несколько). Заголовки могут содержать любую информацию, связанную с приложением. В

теле сообщения указываются запрашиваемый метод с его аргументами либо XML-документ.

SOAP-заголовок и тело вложены в конверт, имеющий вид:

```
<soap-env:Envelope
   xmlns:soap-env="http://www.w3.org/2001/06/soap-envelope">
     <!-- Далее следует заголовок -->
  <soap-env:Header>
...
  </soap-env:Header>
     <!-- Далее следует тело -->
  <soap-env:Body>
. . .
  </soap-env:Body>
</soap-env:Envelope>
```

Прием информации, содержащейся в теле SOAP-сообщения, выполняется SOAP-процессором, который пересылает сведения из тела сообщения в запрошенную службу (например, EJB).

Кроме языка XML и Интернет-протоколов, таких как HTTP, в технологии Web-служб входят следующие компоненты:

- язык WSDL (Web Service Description Language) — язык объявления возможностей (функций) Web-службы, описания на WSDL являются XML-документами;
- стандарт UDDI (Universal Description, Discovery and Integration), служащий для фиксации возможностей Web-службы.

Стандарт UDDI помогает Web-сервис найти, WSDL — его охарактеризовать, а SOAP — взаимодействовать с ним.

Механизм взаимодействия клиента и сервера в технологии SOAP можно представить в виде последовательности следующих процедур (рис. 2):

1. Клиентское приложение создает экземпляр объекта **SOAPClient**;
2. **SOAPClient** просматривает UDDI, тем самым определяется нужный метод Web-сервиса;
3. **SOAPClient** формирует SOAP-сообщение и отправляет его серверу;
4. Серверная программа Listener принимает SOAP-сообщение, создает объект **SOAPServer** и передает ему это сообщение;
5. **SOAPServer** вызывает метод Web-сервиса;
6. Результаты помещаются объектом **SOAPServer** в ответное сообщение и передаются клиенту;

7. Объект **SOAPClient** проводит разбор принятого сообщения и возвращает клиентскому приложению результаты работы Web-сервиса.

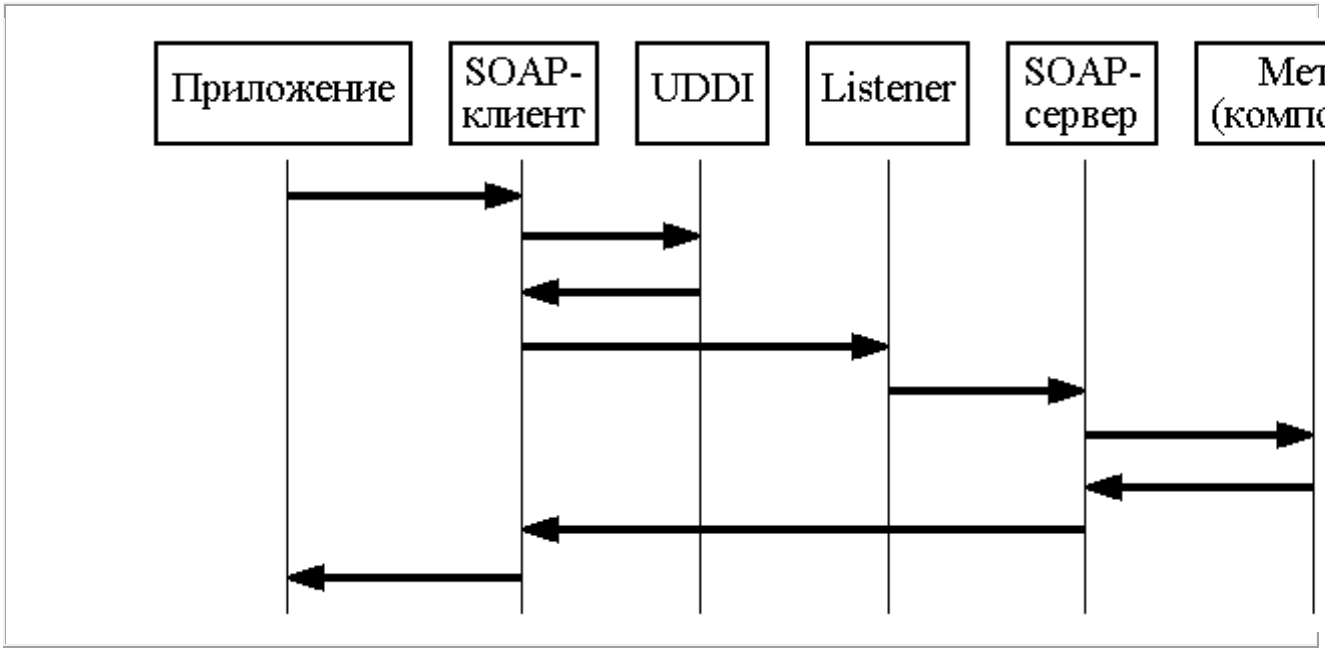


Рис. 2. Взаимодействие клиента и сервера в SOAP

Пример системы интеграции Internet-ресурсов — платформа .NET Framework компании Microsoft. В нее входят семейство .NET Enterprise Servers, представляющее собой набор корпоративных серверных приложений, и визуальная среда VisualStudio.NET, поддерживающая все основные языки программирования и используемая для разработки и потребления Web-сервисов.

Компания IBM предлагает сервер приложений WebSphere Application Server, MQ Series для управления сообщениями для объединения систем, включая поддержку SOAP и Web-сервисов на уровне СУБД DB2.

Для поиска нужных пользователю Web-служб создана спецификация *UDDI* (Universal Description Discovery and Integration - универсальный стандарт описания, обнаружения и интеграции документов), которая является каталогом (аналогичным телефонной книге) Web-служб. Стандарт UDDI служит для описаний возможностей Web-служб, размещаемых в Internet. В описания могут входить названия компаний, контактные реквизиты, URL предоставляемых Web-сервисов, метаданные, описывающие интерфейсы к соответствующим веб-сервисам, и.т.д. Сама служба UDDI-размещена по известному пользователю адресу.

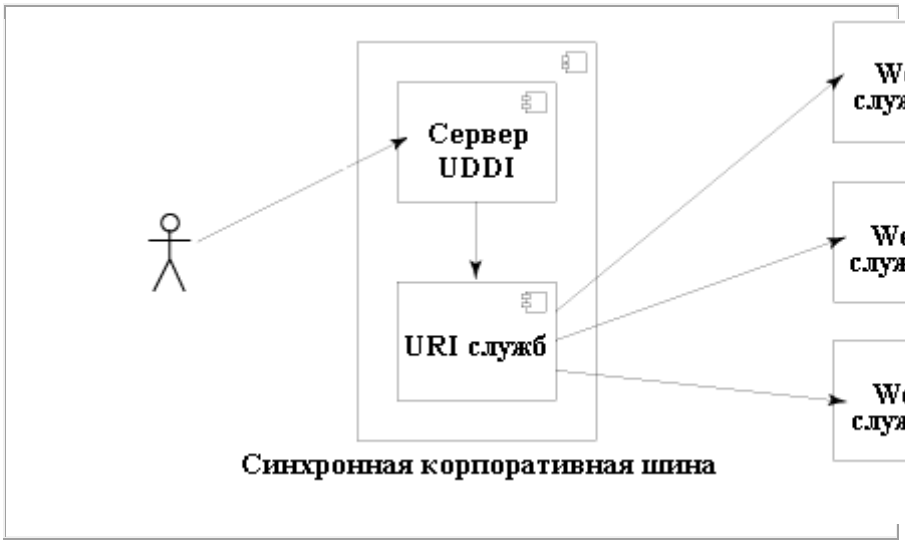


Рис. 1. Архитектура UDDI-службы

Появление стандарта UDDI во-многом связано с потребностями электронной торговли. С его помощью компании могут представлять информацию о своих предложениях ведущим клиентам и торговым партнерам. В стандарте предусмотрены бизнес-реестры, обеспечивающие быстрый и простой доступ к информации об услугах компаний.

UDDI базируется на записях четырех типов: Business Entity, Business Service, Binding Template и Technology Model.

Элемент Business Entity описывает приложение, к которому относится данный Web-сервис. Этот элемент может включать описания категорий приложения для облегчения поиска данных. Это так называемые "Белые страницы", напоминающие телефонную книгу и включающие адреса, контактную информацию и известные идентификаторы компаний.

Элемент Business Service — это класс сервисов в рамках определенного приложения. Это так называемые "Желтые страницы" с информацией об отраслевых кодах, наименованиях продуктов, идентификаторах бизнеса и т.п.

Вместе элементы Binding Template и Technology Model описывают технические характеристики Web-службы. Это "Зеленые страницы", включающие ссылки на спецификации для Web-служб, на различные файлы и механизмы обнаружения, основанные на URL. Важной характеристикой является адрес для вызова данной Web-службы. Адресом может быть URL, адрес электронной почтыили телефонный номер. Т.е. UDDI напоминает по своей архитектуре службу доменных имен DNS.

В большинстве случаев элементы UDDI описываются на языке WSDL, являющемся подмножеством XML.

Технология UDDI включает API для публикаций и для поиска информации.

Организации, заинтересованные в распространении информации, используют API для публикаций. Описания в виде репозитария UDDI (базы данных общего пользования, в которой компании сами себя регистрируют) имеются в сети Internet на определенных серверах (например, на серверах компаний Microsoft, Hewlett Packard, IBM, SAP). Для регистрации в репозитарии, последующей поддержки сведений о себе и извлечения нужной информации в распоряжение компаний стандарт UDDI предоставляет ряд команд (сохранение или удаление той или иной записи, просмотр данных, активизация Web-сервиса, доступного благодаря Binding Template и Technology Model и т.п.).

Для поиска информации UDDI предоставляет ряд программных интерфейсов запроса. В ответах могут содержаться данные об организациях, их услугах, коды доступа к сервису.

Язык WSDL (Web Service Description Language) предназначен для описания Web-служб. В основе WSDL лежит язык XML.

С помощью WSDL представляются типы данных, операции и привязки.

Описание сервиса на языке WSDL выглядит довольно громоздко. Однако это не является осложняющим фактором, поскольку для генерации WSDL-файлов имеются удобные редакторы, примером которых могут служить XML Spy.

В начале WSDL-описания указывается имя сервиса, ссылка на пространство имен WSDL и на целевое пространство имен (targetNamespace), которому будут принадлежать все имена, объявляемые в этом документе.

Далее следуют части WSDL-описания: типы данных, используемых Web-службой (types); сообщения (message) — формат запросов и ответов; типы портов (portType) — набор операций, выполняемых Web-службой; связи (binding) — адрес для вызова операции (service):

```
<definitions
name="имя сервиса"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://...
>
<types> ...
<message> ...
<portType> ...
<binding> ...
<service> ...
```

</definition>

Типы данных, фигурирующие в сообщениях, которыми обмениваются пользователи распределенных приложений, должны правильно интерпретироваться как отправителем, так и получателем. Поэтому типы описываются с помощью XML-схемы, входящей в WSDL-файл, т.е. становятся доступными обоим участникам связи. Пример части types (вместо многоточия должны быть указаны конкретные имена и элементы):

```
<types>
  <xsd:schema
    targetNamespace="http://..."
    xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
     <xsd:complexType>
       <xsd:sequence>
        <xsd:element name="..." type="..."/>
        ...
       </xsd:sequence>
     </xsd:complexType>
     </xsd:element name="..." type="..."/>>
        ...
   </xsd:schema>
</types>
```

Часть message служит для указания данных, которыми будут обмениваться участники связи.

Часть message состоит из одного или более элементов part, где каждый элемент part ассоциирован или с element (при использовании документ-стиля), или с type (при использовании RPC-стиля). В случае документ-стиля базовая структура описания сообщения имеет вид (* означает нуль или более):

```
<message name="..."> *
  <part name="..." element="..."/> *
</message>
```

В случае RPC-стиля в теге part вместо element="..." записывается type="..." и значение атрибута name из тега part становится именем элемента в конкретном сообщении.

Часть portType (иначе интерфейс) определяет группу операций. Операции в WSDL служат для интерпретации данных, содержащихся в сообщениях.

Каждая операция (operation) содержит элементы input и/или output (при наличии элемента output возможен еще элемент fault). Параметр message у

элементов input и output указывает на описание соответствующего сообщения (message).

```
<portType name="...">
...
 <operation name="...">
  <input message="..."/>
  <output message="..."/>
 </operation>
...
</portType>
```

Имеется несколько вариантов операций. Если элемент input предшествует элементу output, то конечный узел получает сообщение и отправляет соответствующий ответ. Такая операция называется запрос-ответ (request-response). Обратный порядок output и input определяет операцию требование-ответа (solicit-response), при которой конечный узел отправляет сообщение и получает соответствующий ответ. При наличии только элемента input имеем однонаправленную операцию — сообщение получает конечный узел. Операция, содержащая только элемент output, означает отправку сообщения конечным узлом.

Связи (привязки) используются для отображения типов данных и операций на транспортный протокол. Атрибут type в теге binding ссылается на конкретный portType (интерфейс). Далее указываются тип сервиса (например, SOAP или MIME), стиль представления (document или RPC), транспортный протокол (атрибут transport). Для каждой операции нужно задать URI получателя сообщения (атрибут soapAction, его значение используется в HTTP-заголовке). Имя в теге operation ссылается на имя операции в части portType.

Элемент soap:body определяет, как части сообщения появляются в SOAP-элементе Body. В случае документ-стиля значение атрибута use есть "literal" и это означает, что тело сообщения будет содержать XML-документ и что части сообщения определяют XML-элементы. Использование стиля RPC в SOAP показывает, что тело будет содержать XML-представление вызова метода и что части сообщения представляют параметры метода (use="encoded").

Структура части binding в случае SOAP, транспортного протокола HTTP и стиля RPC имеет вид:

```
<binding name="..." type="...">
 <soap:binding style="rpc"
   transport=
```

```
    "http://schemas.xmlsoap.org/soap/http"/>
  <operation name="...">
    <soap:operation soapAction="..."/>
      <input>
        <soap:body
         use="encoded"
         encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body
         use="encoded"
         encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
  </operation>
  ...
  <operation name="...">
    <soap:operation soapAction="..."/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
  </operation>
</binding>
```

Часть service может описывать несколько портов для разных транспортных протоколов и служит для задания имени (service name="...") и адреса сервиса (location="http:..."). Атрибут binding указывает на имя, присвоенное помещенной выше части binding.

```
<service name="...">
    <port name="..."
     binding="...">
      <soap:address
       location="http:..."/>
    </port>
</service>
```

Ниже приведен пример, взятый из источника [1], WSDL-документа, описывающего Веб-сервис, предоставляющий всего одну операцию Sum (сложение двух целых чисел).

```
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:s="http://www.w3.org/2001/XMLSchema"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:s0="http://tempuri.org"
             xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
             targetNamespace="http://tempuri.org"
xmlns="http://schemas.xmlsoap.org/wsdl/">
    //Описание типов данных аргументов метода и возвращаемого значения
      <types>
        <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org">
    // Методу Sum передаются два аргумента val1 и val2 с указанными типами данных
          <s:element name="Sum">
            <s:complexType>
              <s:sequence>
                <s:element name="val1" type="s:long" minOccurs="0" />
                <s:element name="val2" type="s:long" minOccurs="0" />
              </s:sequence>
            </s:complexType>
          </s:element>
    // Описание типа данных возвращаемого методом значения
          <s:element name="SumResponse">
            <s:complexType>
              <s:sequence>
                <s:element name="SumResult" type="s:long" minOccurs="0"
/>
              </s:sequence>
            </s:complexType>
          </s:element>
        </schema>
```

```
  </types>
  // Описание входящего сообщения метода Sum с входящим сообщением; ассоциирован
  //тип данных Sum
  <message name="SumSoapIn">
    <part name="parameters" element="s0:Sum" />
  </message>
  // Описание исходящего сообщения метода Sum с исходящим сообщением; ассоциирован тип данных
  //  SumResponse
  <message name="SumSoapOut">
    <part name="parameters" element="s0:SumResponse" />
  </message>
  // Описание операций (методов), предоставляемых Веб-сервисом
  <portType name="ArithmeticSoap">
  // Данный Веб-сервис предоставляет операцию Sum, операция имеет входящее сообщение SumSoapIn
  // и исходящее сообщение SumSoapOut
     <operation name="Sum">
       <input message="s0:SumSoapIn" />
       <output message="s0:SumSoapOut" />
     </operation>
  </portType>
  // Определение формата сообщения и деталей протокола для каждого порта
  <binding name="ArithmeticSoap" type="s0:ArithmeticSoap">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
     <operation name="Sum">
       <soap:operation soapAction="http://tempuri.org/Web.Arithmetic.Sum" style="document" />
       <input>
         <soap:body use="literal" />
       </input>
       <output>
         <soap:body use="literal" />
       </output>
    </operation>
```

```
  </binding>
  // Определяет имя сервера Веб-служб, позволяет объединить внутри себя несколько портов
  // (наборов методов), определяет расположение сервиса
  <service name="Arithmetic">
    <port name="ArithmeticSoap" binding="s0:ArithmeticSoap">
      <soap:address
location="http://MASHA:1972/csp/www/Web.Arithmetic.cls" />
    </port>
  </service>
</definitions>
```

К числу средств интеграции MCAD среднего уровня с ERP системами (*интеграции MCAD и ERP*) относятся:

- SolidWorks-ERP Bridge компании BWIR. Программа позволяет искать, создавать и обновлять информацию о сборках и деталях. Благодаря своей открытой архитектуре, может быть использована для интеграции любой САПР с любой ERP-системой.
- AgniLink компании Elmo.
- CAD Desktop компания Cideon. Предназначено для интеграции Inventor и SAP ERP.

Microsoft *BizTalk Server* — комплексная система интеграции приложений и создания схем ведения бизнеса. Назначение системы — описывать документы разных форматов, преобразовывать документы из одного формата в другой, доставлять их по разным транспортным протоколам, отслеживать маршруты следования всех документов из одной компании в другую и управлять всеми связями между приложениями. В структуре Microsoft BizTalk Server имеется несколько подсистем: BizTalk Editor, BizTalk Mapper, BizTalk Messaging Manager, BizTalk Server Administration, BizTalk Orchestration Designer и др.

1. BizTalk Editor предназначен для создания спецификаций бизнес-документов. Для пересылки любых данных применяется XML, но система позволяет работать и с другими форматами, напримерEDIFACT. Документ нужно прежде всего описать. Для этого необходимо создать спецификацию, т.е. структуру бизнес-документа в BizTalk-редакторе.

2. BizTalk Mapper — визуальное средство для установления связей между полями документов различных форматов. BizTalk Mapper обеспечивает преобразование документов из одного формата в другой с использованием формата XSLT (Extensible Stylesheet Language Transformations). С помощью

BizTalk Mapper может производиться и обработка данных с помощью специализированных функций.

3. BizTalk Messaging Manager предназначен для управления обменом документами. BizTalk Server поддерживает: стандарты X12 (ANSI) и EDIFACT (ООН) для документооборота, протоколы HTTPи HTTPS, SMTP, а также обмен табличными (flat) файлами и отправку документов по факсу. Компании-имеют возможность создавать собственные программы-адаптеры, обеспечивающие их приложения возможностью прямого обмена документами. С помощью BizTalk Messaging Manager определяются характеристики организаций (торговых партнеров) и приложений, а также определения документа (document definitions) и некие метаданные документа, называемые конвертами (envelopes). Home Organization представляет собственную организацию.

4. BiTalk Server Administration — средство администрирования сервера (конфигурирование серверов и групп, управление очередями и базами данных, обеспечение трекинга, защита данных, шифрование и пр.).

5. Microsoft BizTalk Server обеспечивает интеграцию приложений как внутри организации, так и между организациями независимо от форматов, в которых представлены документы.

6. BizTalk Orchestration Designer предназначен для формирования и реализации бизнес-процесса. Визуальные средства BizTalk Orchestration Designer обеспечивают бизнес-аналитиков, специалистов по информационным технологиям (IT) и разработчиков универсальной средой для совместного проектирования и построения динамических, распределенных в пространстве бизнес-процессов.

7. BizTalk Management Desk — инструмент конфигурирования бизнес-параметров (формирование договоров с контрагентами и поддержка рассылки документов с учетом их содержания по определенным правилам);

8. Набор служб для обработки входящих и исходящих документов:

- поддержка транспортных сетевых протоколов, таких, как HTTP, HTTPS, FTP, SMTP, SMB (передача файлов), Microsoft Message Queue Server (MSMQ) и Microsoft Exchange;
- обеспечение надежной защиты — поддержка шифрования, цифровых подписей и инфраструктуры с использованием открытого ключа (public key infrastructure);
- трекинг документов для контроля за их движением;
- средства анализа, позволяющие исследовать бизнес-процессы и генерировать любые отчеты, а также поддержка встроенных в Microsoft Office средств анализа данных.

9. Репозиторий для хранения схем и документов.

10. Набор готовых компонентов для обработки документов. В частности, сейчас имеются адаптационные программы, обеспечивающие прямую интеграцию XML в системы SAP R/3 и EDI. Расширяемая архитектура позволит интегрировать XML и с другими бизнес-приложениями.

11. Набор средств для разработчика (SDK).

Интеграция с Microsoft SQL Server обеспечивает расширенные возможности оперативной аналитической обработки (OLAP).

Основы методологий проектирования автоматизированных систем обработки информации и управления

Самара, 2009

УДК 681.3

Рецензенты:

заведующий кафедрой информационных систем и технологий Самарского государственного аэрокосмического университета, д.т.н., профессор С.А. Прохоров

доцент кафедры экономических информационных систем Поволжского государственного университета телекоммуникаций и информатики, к.т.н., доцент А.Р. Диязитдинова

Основы методологий проектирования автоматизированных систем обработки информации и управления / Учебное пособие / Составитель: А.В. Иващенко, Самара: СНЦ РАН, 2009 – 40 с., ил.

ISBN – 978-5-93424-421-8

Данное пособие содержит фрагменты конспекта лекций по курсу «Проектирование автоматизированных систем обработки информации и управления».
Пособие предназначено для студентов специальности 230102 «Автоматизированные системы обработки информации и управления».

Печатается по решению издательского совета
Самарского научного центра Российской академии наук

ISBN – 978-5-93424-421-8

СОДЕРЖАНИЕ

Введение

Системный подход к проектированию и разработке автоматизированной системы обработки информации и управления (АСОИУ) заключается, в основном, в моделировании и всестороннем анализе требований к этой системе. Под моделированием при этом понимается процесс создания достаточно точного и адекватного графического описания системы, а также интерпретация полученного описания для определения оценочных значений ее некоторых характеристик.

Модель (model) – это искусственный объект, представляющий собой отображение (образ) системы и ее компонентов. Модель может описывать существующие (AS-IS), идеализированные (SHOULD-BE) и вновь создаваемые (TO-BE) процессы и функции. На разных стадиях разработки используются различные уровни детализации модели.

При необходимости, модель может быть создана с использованием обычных графических и текстовых редакторов. Для обеспечения согласованности модели и ее высокой степени формализации, которая требуется при описании технического решения, необходимо использовать современные технологии автоматизированного проектирования – CASE-технологии (Computer Aided Software/System Engineering).

CASE-технология представляет собой методологию проектирования, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область и производить ее анализ на всех этапах разработки и сопровождения системы.

Данное пособие содержит краткое описание основных подходов к проектированию АСОИУ, изложенных в наиболее распространенных методологиях (см. Таблицу 1) и предназначено для использования в качестве конспекта части лекций по курсу «Проектирование АСОИУ». Для более глубокого изучения в списке литературы указаны соответствующие источники.

Таблица 1. Соответствие понятий проектирования АСОИУ.

	Методология	**Нотация или язык**	**Стандарт**	**CASE средства**
	Общие методы и технологии проектирования	Правила построения диаграмм	Формализа-ция подходов и языка	Инструмент автоматизирован-ного построения
Соответствие	Функциональная	SADT	IDEF0	BPWin (AllFusion), IDEF Doctor, MS Visio
	Потоков данных	DFD	–	
	Процессная	–	IDEF3	
	Сущность-связь	ER диаграммы	IDEF1X	ERWin (AllFusion)
	Объектно-ориентированная	UML	UML 2.0	Rational Rose, Star UML, Magic Draw MS Visio
	Процессная	eEPC/PCD, VAD	ARIS	ARIS Toolset, MS Visio

1 Методология SADT. Стандарт IDEF0

SADT (Structured Analysis and Design Technique, методология Росса) – методология структурного анализа и проектирования, основанная на графическом представлении системы разными языковыми средствами. Основные положения данной методологии зафиксированы в стандарте IDEF0.

В IDEF0 [1 – 3] система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Процесс моделирования системы в IDEF0 начинается с определения контекста, то есть наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта (области) моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно определить, что входит в систему, а что лежит за ее пределами.

Точка зрения (viewpoint) – указание на должностное лицо, с позиции которого разрабатывается модель. У модели может быть только одна точка зрения. Изменение точки зрения приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения.

Формулировка цели (purpose) выражает причину создания модели, то есть содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру.

Область моделирования (scope) описывает круг функций системы. При формулировании области необходимо учитывать два компонента: широту и глубину. Широта подразумевает определение границ моделирования, а глубина определяет, на каком уровне детализации модель является завершенной.

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм, разработанных с определенной целью и с выбранной точки зрения.

Каждая такая диаграмма содержит работы и стрелки.

Работы или функциональные блоки (activity) обозначают поименованные процессы, функции или задачи, которые выполняются в течение определенного времени и имеют распознаваемые результаты. Название функционального блока – это глагол или глагольный оборот.

На каждой диаграмме отображается от трех до шести работ. Работы имеют доминирование – они размещаются на диаграмме по степени важности. Самой доминирующей работой диаграммы может быть либо первая из требуемой последовательности, либо планирующая или управляющая. Наиболее доминирующая работа располагается в левом верхнем углу диаграммы, наименее доминирующая – в правом нижнем. Работы на одной диаграмме нумеруются последовательно в порядке их доминирования.

Взаимодействие работ с внешним миром и между собой описывается в виде стрелок. Стрелка (arrow) – направленная линия, состоящая из одного или нескольких сегментов, которая моделирует канал, передающий данные от источника (начальная точка стрелки) к потребителю (конечная точка с «наконечником»).

В IDEF0 различают пять типов стрелок:

- вход (input) – материал или информация, которые используются или преобразуются работой для получения результата. Допускается, что работа может не иметь ни одной стрелки входа. Входные стрелки рисуются как входящие в левую грань работы;
- выход (output) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Выходные стрелки рисуются как исходящая из правой грани работы;
- управление (control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелки управления рисуются как входящие в верхнюю грань работы;
- механизм (mechanism) – ресурсы, которые выполняют работу. Стрелки механизма рисуются как входящие в нижнюю грань работы;
- вызов (call) – специальная стрелка, указывающая на другую модель. Стрелка вызова рисуется как исходящая из нижней грани работы.

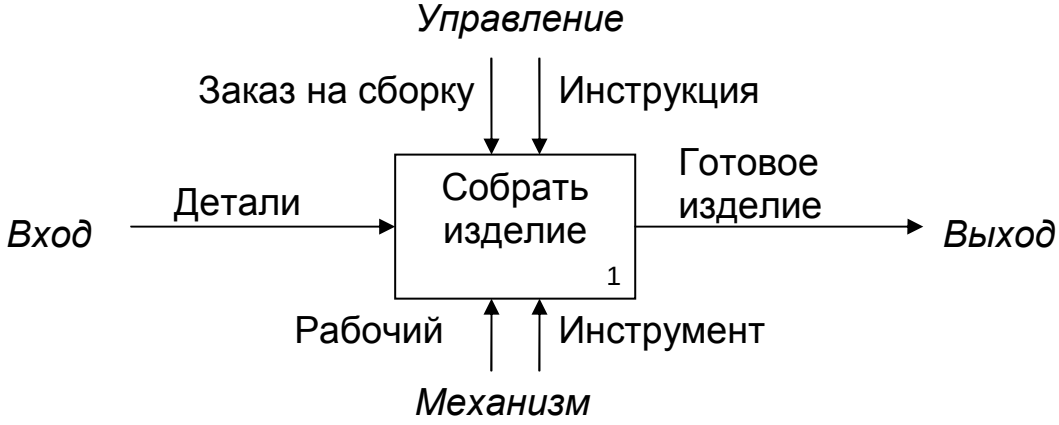


Рис. 1. Типы стрелок

Все работы на диаграмме являются преобразующими, то есть преобразуют входы в выходы под действием управлений при помощи механизмов.

Стрелка обычно представляет набор объектов. Поэтому они могут разветвляться и соединяться различными способами. Сегменты стрелок, за исключением стрелок вызова, помечаются существительным или оборотом существительного.

Внутренние стрелки используются для связи работ между собой. В IDEF0 требуется только пять типов взаимосвязей между блоками с помощью стрелок для описания их отношений: вход, управление, обратная связь по входу, обратная связь по управлению, выход-механизм (см. рис. 2).

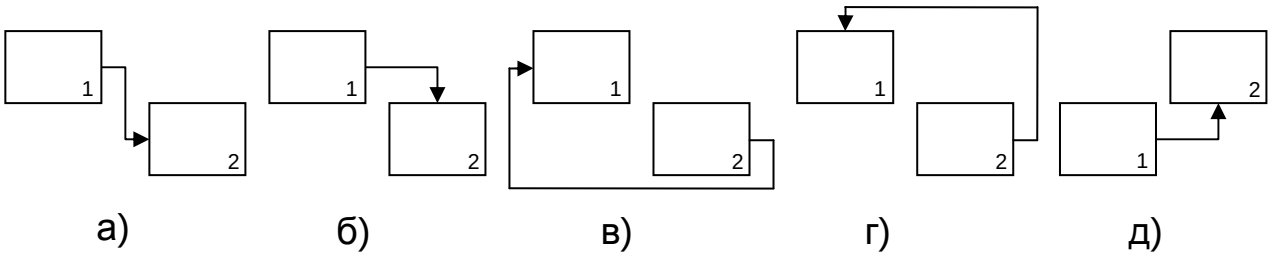


Рис. 2. Типы взаимосвязи между работами: а) вход, б) управление, в) обратная связь по входу, г) обратная связь по управлению, д) выход-механизм

Следует отметить ряд особенностей построения диаграмм в IDEF0, связанных с использованием стрелок управления:

- в связи с тем, что поступление данных на вход функционального блока само по себе не является причиной их обработки, вход всегда должен быть сопровожден соответствующим управлением;
- главное различие между управлением и входом заключается в том, что входные данные преобразуются, а управление либо регламентирует преобразование, либо инициирует его;
- в случае, когда данные входа малозначительны либо разделение входа и управления не обосновано, следует использовать управление;
- регламентирующее управление (стандарты, регламенты и т.п.) используется для определения требований к функции – для начала ее выполнения обязательно нужно инициирующее управление;
- поскольку одна диаграмма IDEF0 часто содержит описание нескольких случаев последовательного выполнения функций, рекомендуется строить диаграммы так, чтобы можно было по управлению проследить каждую последовательность.

Каждая работа (функциональный блок) может быть разбита на более мелкие работы, взаимосвязанные между собой. Этот процесс называется функциональной декомпозицией (decomposition), а диаграммы, которые описывают каждую работу (функциональный блок) и взаимодействие работ, называются диаграммами декомпозиции.

Вершиной древовидной структуры взаимосвязанных диаграмм является контекстная диаграмма, которая имеет шифр А-0. Контекстная диаграмма (context diagram) представляет собой самое общее описание системы и ее взаимодействия с внешней средой. Контекстная диаграмма содержит только один функциональный блок, соединенный с границами диаграммы при помощи граничных стрелок.

Граничные стрелки служат для описания взаимодействия системы с окружающим миром. Идентификация граничных стрелок осуществляется с помощью ICOM-кодов, которые содержат префикс, соответствующий типу стрелки (Input, Control, Output, Mechanism).

После описания системы в целом на контекстной диаграмме в виде одного функционального блока, производится его разбиение на крупные

фрагменты. Диаграмма, декомпозирующая работу на контекстной диаграмме называется диаграммой декомпозиции верхнего уровня и имеет шифр А0.

Диаграммы декомпозиции функциональных блоков диаграммы А0 в соответствии с номерами блоков, проставленных по порядку доминирования, будут иметь шифр А1, А2 и т.д. Номер каждой последующей диаграммы декомпозиции состоит из номера родительской диаграммы и номера блока (например, А121).

Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования. Наиболее важные свойства системы обычно выявляются на верхних уровнях иерархии и уточняются по мере декомпозиции.

Кроме контекстной диаграммы и диаграмм декомпозиции, модель IDEF0 может содержать диаграмму дерева узлов и диаграмму только для экспозиции (For Exposition Only, FEO). Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмма для экспозиции строится для иллюстрации отдельных фрагментов модели, альтернативной точки зрения, либо для специальных целей.

Рассмотрим пример модели IDEF0 для задачи выполнения заказов. Цель моделирования: Повысить оперативность приема заказов за счет внедрения АСОИУ. Точка зрения: Руководитель отдела обслуживания клиентов.

Контекстная диаграмма А-0 (см. рис. 3) показывает, что ведение учета заказов предусматривает обработку параметров заказов и данных заказчиков при поступлении новых заказов и информационных запросов, а также в случае отмены заказов. При этом система выводит сообщения об изменении заказов и сведения об обработанных заказах.

Диаграмма декомпозиции А0 (см. рис. 4) иллюстрирует ход обработки заказов. При поступлении нового заказа его параметры обрабатываются в блоке «Принимать заказы». В случае если данный заказчик обратился впервые, происходит обработка его данных в блоке «Вести учет заказчиков» по управлению «Сообщение о новом заказчике». Данные о событии «Отмена заказа» поступают в качестве управления в блок «Контролировать исполнение», в результате чего в блок «Принимать заказы» передается соответствующий запрос и данные отмененного заказа.

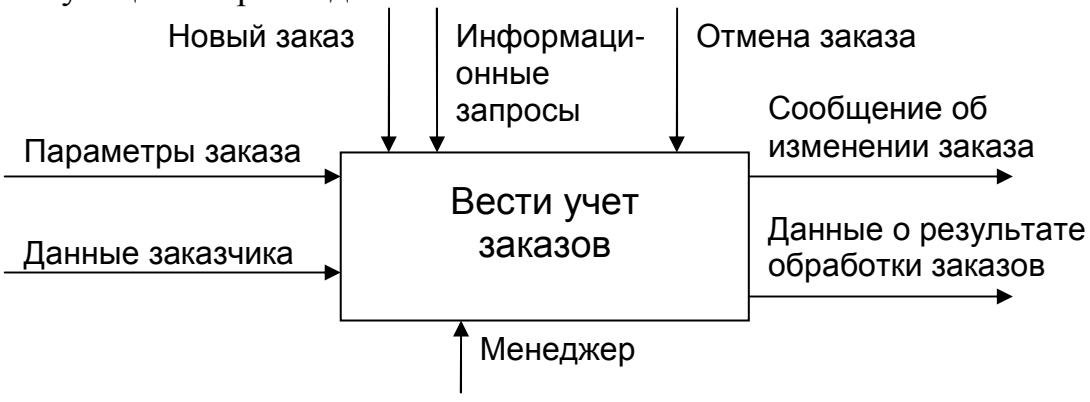


Рис. 3. Контекстная диаграмма А-0

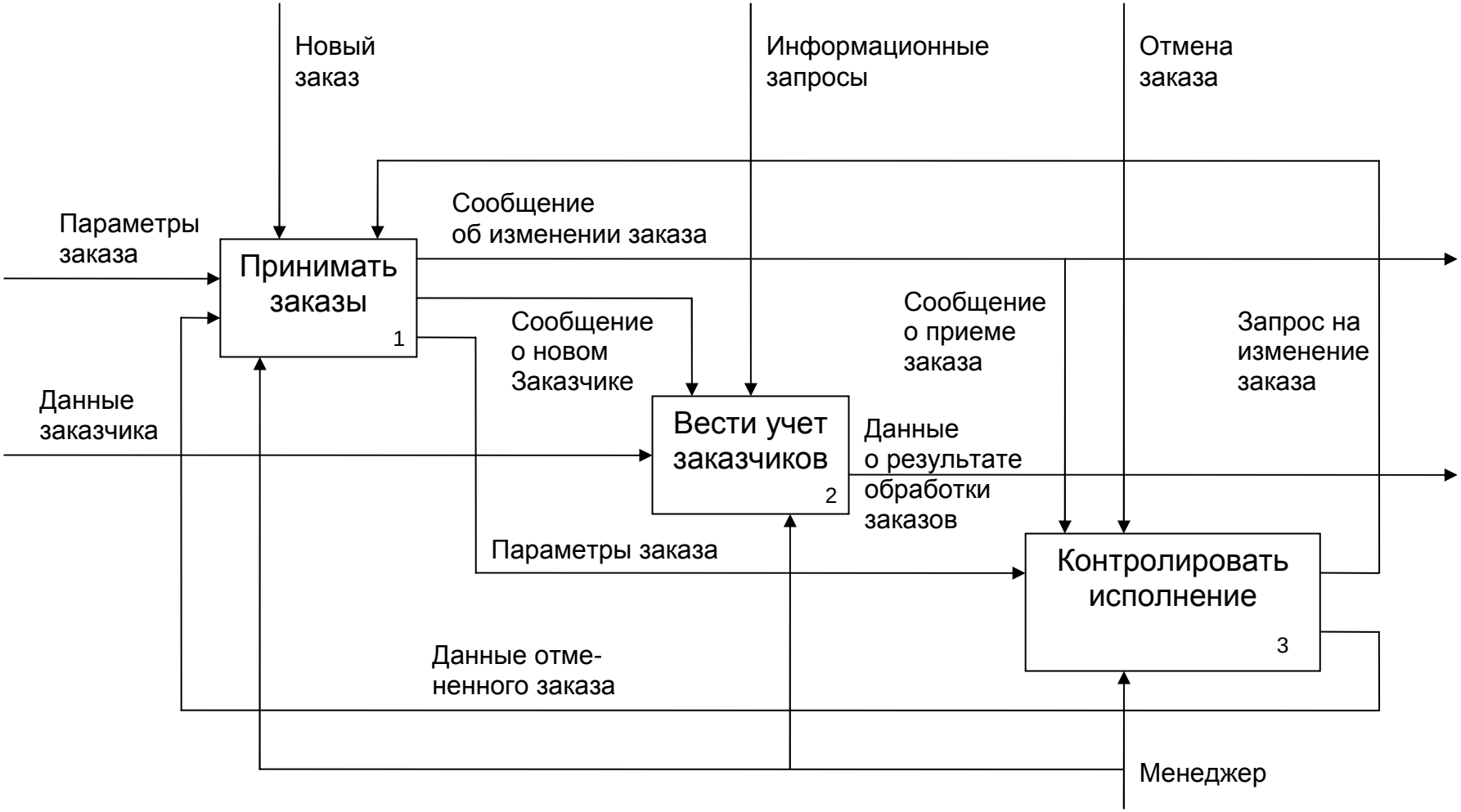


Рис. 4. Диаграмма декомпозиции верхнего уровня A0

2 Диаграммы потоков данных DFD

Диаграммы потоков данных (Data Flow Diagrams, DFD) описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией как внутри системы между бизнес-функциями, так и системы в целом с внешней информационной средой. Модель DFD [1 – 2] представляет собой набор диаграмм, отражающих различные аспекты модели, с учетом выбранной точки зрения и цели моделирования.

Работы в DFD представляют собой функции системы, преобразующие входную информацию в выходную в соответствии с действиями, задаваемыми именами работ. Каждая работа имеет уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса работы во всей модели.

Внешние сущности (ссылки) указывают на место, организацию или человека, которые участвуют в процессе обмена информацией с системой, но располагаются за рамками данной модели. Внешняя сущность является источником или приемником данных извне модели. Внешние сущности обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах с целью повышения наглядности.

Потоки данных используются для моделирования передачи информации (или физических компонентов) из одной части системы в другую. Потоки изображаются на диаграмме именованными стрелками, ориентация которых указывает направление движения информации. Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы.

В DFD также применяются двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и внешней сущностью и между внешними сущностями. В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя. В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой.

Хранилище данных – это место накопления информации внутри системы. Хранилища данных позволяет на определенных участках определять данные, которые будут сохраняться в памяти между работами. В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое.

Фактически хранилище представляет «срезы» потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке. Хранилище данных может содержать информацию длительного хранения или временную информацию. Имя хранилища должно

идентифицировать его содержимое. На одной диаграмме может присутствовать несколько копий одного и того же хранилища данных.

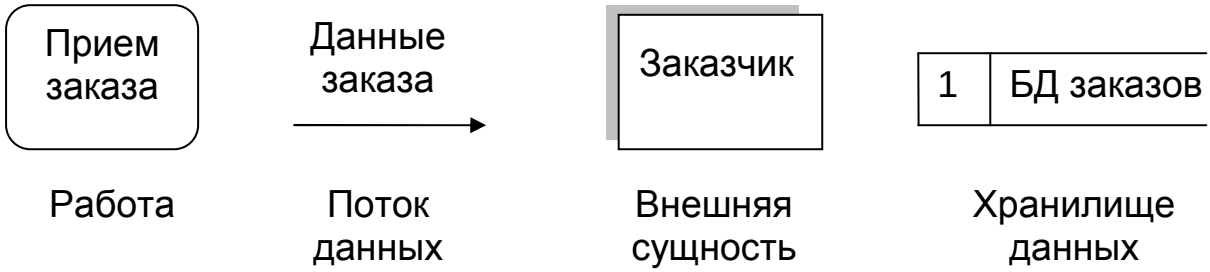


Рис. 5. Графические примитивы DFD

Построение диаграмм DFD производится «сверху вниз» в соответствии с целью моделирования и точкой зрения. Сначала строится диаграмма контекста, описывающая исследуемую систему целиком. Затем следует диаграмма первого уровня для описания наиболее работ. При этом внешние сущности дублируются на диаграмме декомпозиции, а работы разбиваются. Дальнейшее уточнение может при необходимости производиться с помощью более подробных диаграмм на следующих уровнях.

Результатом моделирования является набор диаграмм, отражающих различные аспекты модели, которая объединяет в себе одну или несколько диаграмм потоков данных.

Пример модели DFD для задачи выполнения заказов приведен на рис. 6 – 7. Цель моделирования: Повысить оперативность приема заказов за счет внедрения АСОИУ. Точка зрения: Руководитель отдела обслуживания клиентов.

Данная модель иллюстрирует процесс выполнения заказа. После получения заказа производится сохранение его параметров, выбор соответствующего товара и выставление счета. После получения данных об оплате счета, производится отгрузка товара. Ведение базы данных товаров производится менеджером. Директору формируется отчет об обслуживании заказчиков.

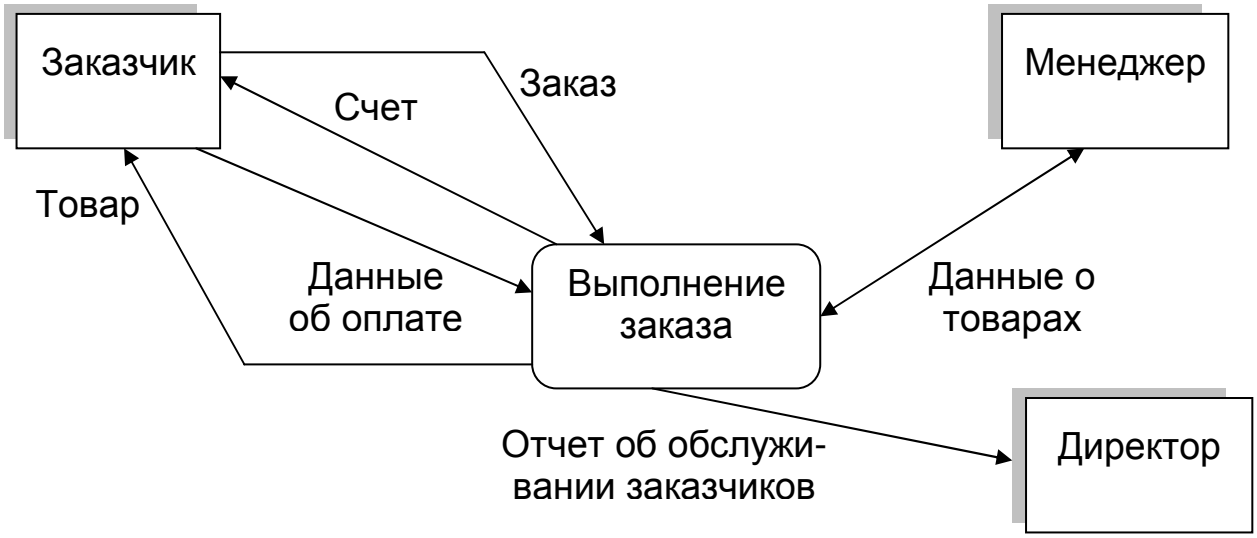


Рис. 6. Диаграмма контекста DFD

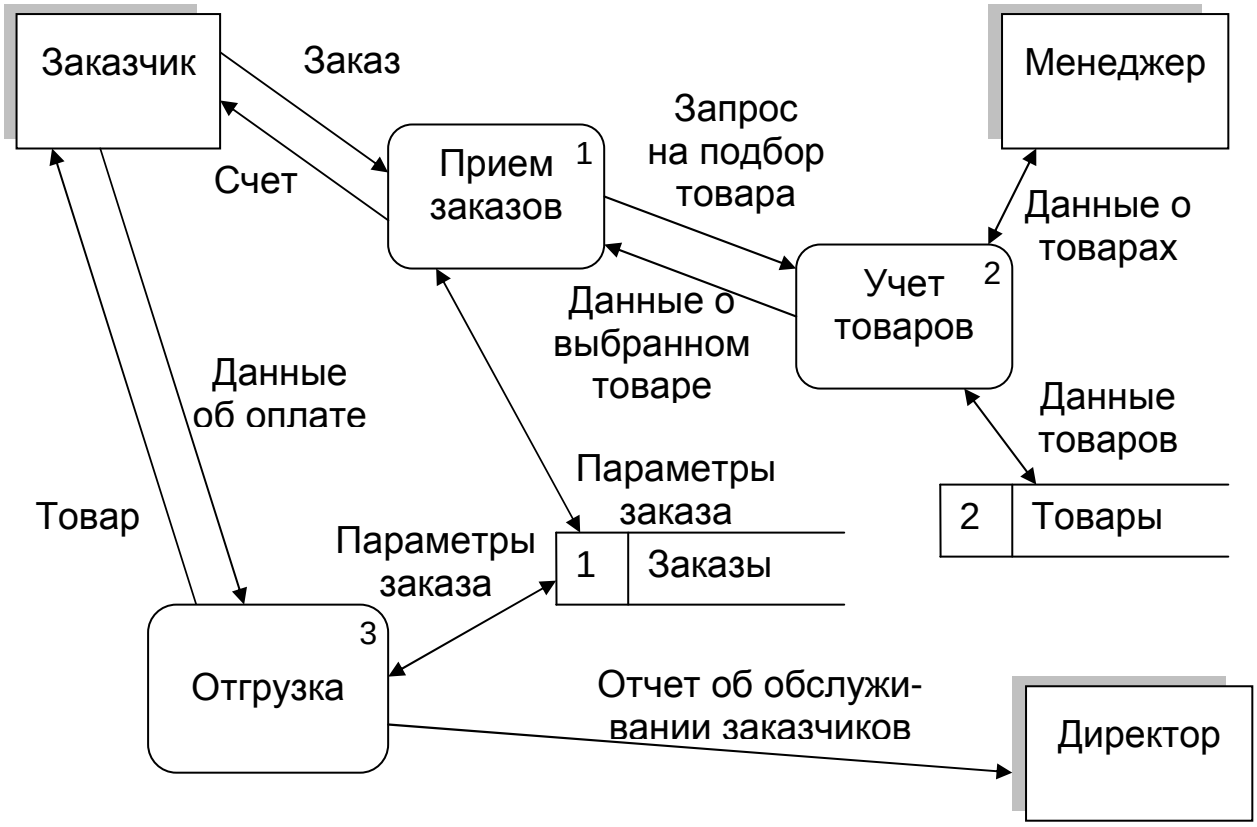


Рис. 7. Диаграмма декомпозиции DFD

3 Моделирование бизнес-процессов. Стандарт IDEF3

Методология моделирования IDEF3 (workflow diagramming) предназначена для описания логики взаимодействия работ и последовательности их выполнения. Диаграммы IDEF3 [1 – 2] позволяют сфокусировать внимание на течении процессов и на отношениях процессов и важных объектов, являющихся частями этих процессов.

Каждая диаграмма в IDEF3 описывает какой-либо сценарий бизнес-процесса – последовательность изменений свойств объекта, которые необходимо выполнить за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования. Моделирование начинается с формулировки точки зрения, цели и области моделирования.

Единицы работы (Unit of Work, UOW), также называемые работами (activity), являются центральными компонентами модели. В IDEF3 работы имеют имя, выраженное отглагольным существительным, обозначающим процесс действия, одиночным или в составе фразы, и номер (идентификатор). Другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы. Идентификатор работы присваивается при создании и не меняется никогда. Даже если работа будет удалена, ее идентификатор не должен вновь использоваться для других работ. Обычно номер работы состоит из номера родительской работы и порядкового номера на текущей диаграмме.

Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи:

- связь предшествования (Precedence) – показывает, что прежде чем начнется работа-приемник, должна завершиться работа-источник;
- связь отношения (Relational) – показывает связь между двумя работами или между работой и объектом ссылки;
- поток объектов (Object Flow) – показывает участие некоторого объекта в двух или более работах, как, например, если объект производится в ходе выполнения одной работы и потребляется другой работой.

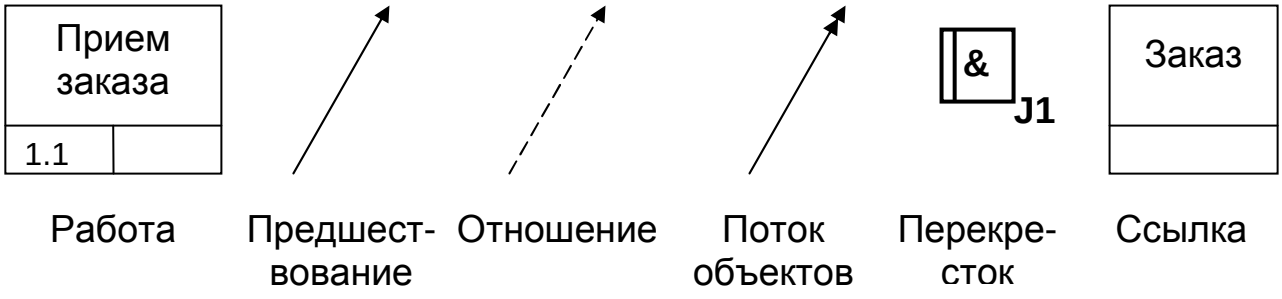


Рис. 8. Графические примитивы IDEF3

Имя стрелки должно ясно идентифицировать отображаемый объект. Связь предшествования показывает, что работа-источник заканчивается ранее, чем начинается работа-цель. Если результатом работы-источника становится объект, необходимый для запуска работы-цели, используется стрелка потока объектов. Отношение показывает, что работа-источник не обязательно должна закончиться, прежде чем работа-цель начнется; более того, работа-цель может закончиться прежде, чем закончится работа-источник.

Рис. 9. Временная диаграмма последовательности выполнения работ

Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Перекрестки применяются, когда окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ.

Различают два типа перекрестков:

- перекресток слияния (Fan-in Junction) – узел, собирающий множество стрелок в одну, указывая на необходимость условия завершенности работ-источников стрелок для продолжения процесса;
- перекресток ветвления (Fan-out Junction) – узел, в котором единственная входящая в него стрелка ветвится, показывая, что работы, следующие за перекрестком, выполняются параллельно или альтернативно.

В IDEF3 стрелки могут сливаться и разветвляться только через перекрестки. Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J. Перекресток не может использоваться одновременно для слияния и для разветвления.

На одной диаграмме IDEF3 может быть создано несколько перекрестков различных типов. Определенные сочетания перекрестков для слияния и разветвления могут приводить к логическим несоответствиям. Чтобы избежать конфликтов, необходимо соблюдать следующие правила:

1. Каждому перекрестку для слияния должен предшествовать перекресток для разветвления;
2. Перекресток, имеющий одну стрелку на одной стороне, должен иметь более одной стрелки на другой.

3. Перекресток для слияния «И» не может следовать за перекрестком для разветвления типа синхронного или асинхронного «ИЛИ», типа исключающего «ИЛИ»;
4. Перекресток для слияния типа исключающего «ИЛИ» не может следовать за перекрестком для разветвления типа «И»;

Таблица 2. Типы перекрестков

Обозначение	Наименование	Пояснение	
		при слиянии стрелок	при разветвлении стрелок
	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronous OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Объект ссылки (Referent) в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой. В качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или перекрестками пунктирными линиями. При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки.

Методология IDEF3 позволяет декомпозировать работу многократно, то есть работа может иметь множество дочерних работ. Это позволяет в одной модели описать альтернативные потоки. Декомпозиция может быть сценарием или описанием. Описание включает все возможные пути развития процесса. Сценарий является частным случаем описания и иллюстрирует только один путь реализации процесса.

Возможность множественной декомпозиции предъявляет дополнительные требования к нумерации работ. Так, номер работы может состоять из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме. Для описания номер декомпозиции равен единице. Для сценария номер декомпозиции всегда больше единицы.

При создании сценария или описания необходимо придерживаться дополнительных ограничений. В сценарии или описании может существовать только одна точка входа. За точкой входа следует работа или перекресток. Для описания может существовать только одна точка выхода. Сценарий может иметь несколько точек выхода.

IDEF3 позволяет внести информацию в модель различными способами. Например, логика взаимодействия может быть отображена графически в виде комбинации перекрестков. Та же информация может быть отображена в виде объекта ссылки типа ELAB в случае, если комбинация перекрестков занимает значительное место и затрудняет расположение работ на диаграмме.

Таблица 3 – Типы объектов ссылки

Тип	Цель описания
OBJECT	Описывает участие важного объекта в работе
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ), возможно на текущей диаграмме, но не обязательно. Если все работы цикла присутствуют на текущей диаграмме, цикл может также изображаться стрелкой, возвращающейся на стартовую работу. GOTO может ссылаться на перекресток
UOB (Unit of behavior)	Применятся, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла. Например, работа «Контроль качества» может быть использована в процессе «Изготовления изделия» несколько раз, после каждой единичкой операции. Обычно этот тип ссылки не используется для моделирования автоматически запускающихся работ
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. NOTE является альтернативой внесению текстового объекта в диаграмму
ELAB (Elabora-tion)	Используется для усовершенствования графиков или их более детального описания. Обычно употребляется для детального описания разветвления и слияния стрелок на перекрестках

Пример модели IDEF3 для задачи выполнения заказов приведен на рис. 10. Цель моделирования: Повысить оперативность приема заказов за счет внедрения АСОИУ. Точка зрения: Руководитель отдела обслуживания клиентов.

Данная диаграмма иллюстрирует процесс обработки заказа. После того, как заказ принят, происходит сохранение данных о заказе и заказчике (если этой информации в базе данных нет). После чего производится выбор товара, и в случае оплаты – отгрузка, а в противном случае – отмена заказа.

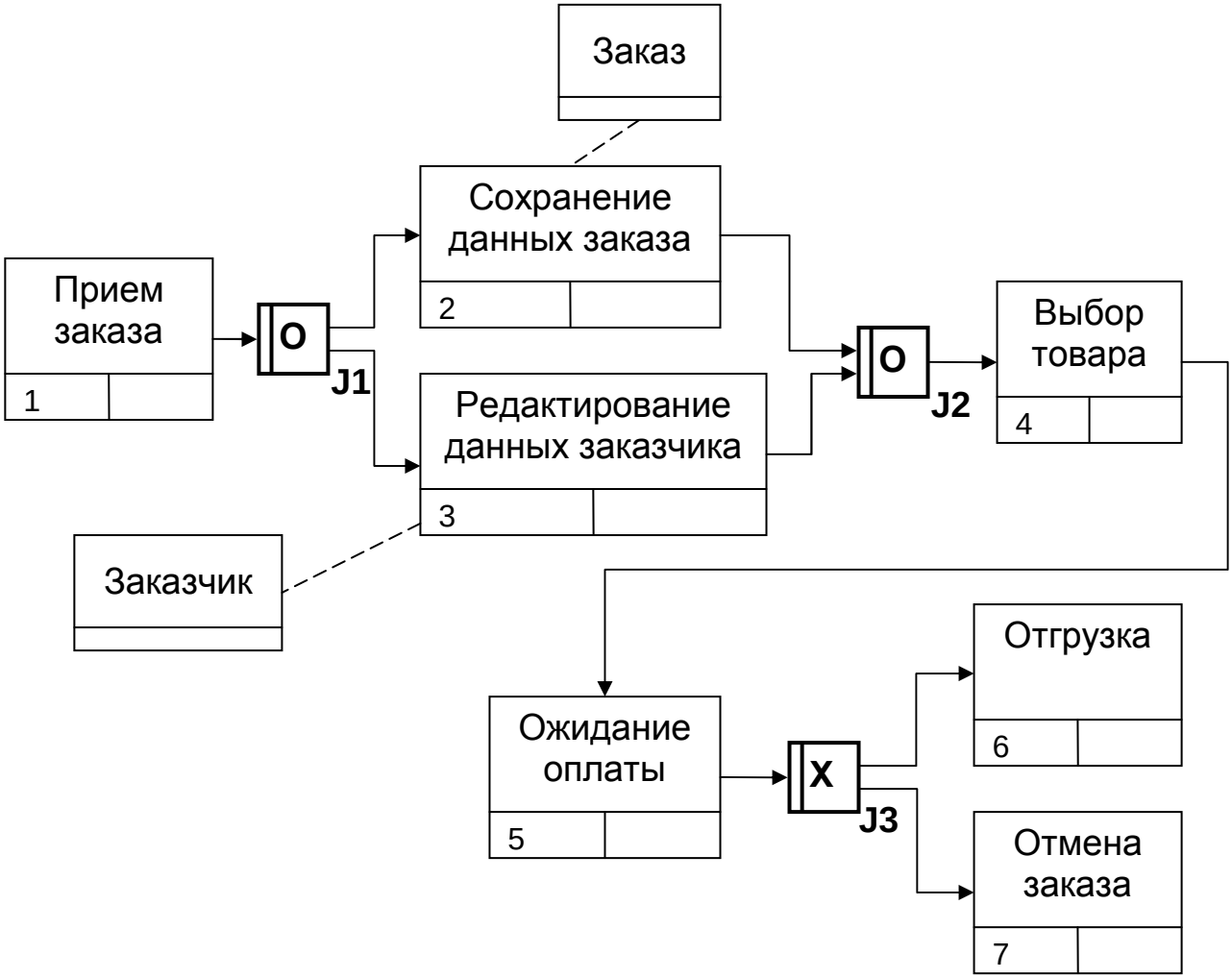


Рис. 10. Процесс IDEF3

4 Построение ER модели данных. Стандарт IDEF1X

Методология IDEF1X [1 – 2] предназначена для моделирования структур баз данных на основе диаграмм сущность-связь (ER-диаграмм). При проектировании баз данных обычно выделяют три уровня абстракции, на которых происходит последовательное уточнение модели: концептуальный (семантический уровень представления данных в виде абстрактных понятий, учитывающих особенности предметной области), логический (уровень представления в виде структуры данных – сущностей, атрибутов и связей) и физический (уровень реализации базы данных). IDEF1X предусматривает проектирование модели базы данных на логическом и физическом уровнях.

Обычно модель одной базы данных описывают с помощью одной или нескольких диаграмм IDEF1X, содержащих сущности, атрибуты и связи.

Сущность определяется как объект, событие или концепция, информация о котором должна сохраняться. Сущности должны иметь наименование (с четким смысловым значением) в виде существительного в единственном числе. Каждый экземпляр сущности на диаграмме уникален.

Атрибут хранит информацию об определенном свойстве сущности. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Атрибут или группа атрибутов, которые однозначно идентифицируют экземпляры сущности, называются первичным ключом (Primary key).

Связь описывает логическое соотношение между сущностями. Каждая связь должна именоваться глаголом или фразой, однако имя связи на диаграмме может не указываться.

На логическом уровне можно установить следующие связи между сущностями: идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим. Связь сущности с другими сущностями определяет ее тип: в IDEF1X различают два типа сущностей: зависимые и независимые.

Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Зависимая сущность не может существовать самостоятельно – экземпляр зависимой сущности определяется только через отношение к сущности, которая его идентифицирует. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности.

Операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ (FK, Foreign Key).

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых атрибутов дочерней сущности.

Связь многие-ко-многим существует только на логическом уровне. При переходе на физический уровень это отношение должно быть преобразовано: вместо связи многие-ко-многим добавляется новая зависимая сущность, свя-

занная идентифицирующими связями один-ко-многим с сущностями, находившимися в исходном отношении.

Мощность связи (Cardinality) – служит для обозначения отношения числа экземпляров одной сущности к числу экземпляров другой. Мощность связи:

- не помечается, когда одному экземпляру родительской сущности соответствуют 0,1 или много экземпляров дочерней сущности;
- помечается символом P, когда одному экземпляру родительской сущности соответствует 1 или много экземпляров дочерней сущности (исключено нулевое значение);
- помечается символом Z, когда одному экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности (исключены множественнные значения);
- помечается цифрой при точном соответствии, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

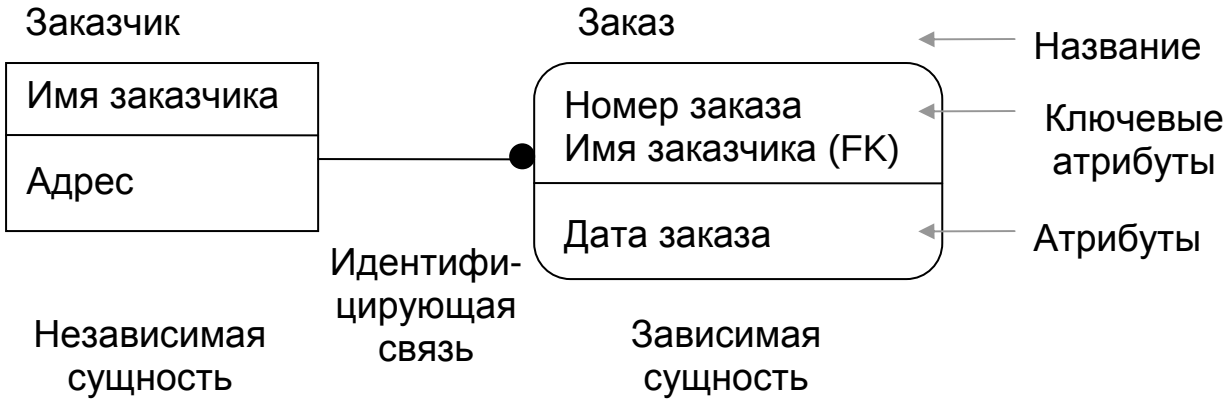


Рис. 11. Графические примитивы IDEF1X. Зависимые и независимые сущности

Для неидентифицирующей связи можно указать обязательность. В случае обязательной связи (No Nulls) при генерации схемы базы данных атрибут внешнего ключа получит признак NOT NULL, несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи (Nulls Allowed) внешний ключ может принимать значение NULL. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности.

Иерархия наследования (категорий) представляет собой особый тип объединения сущностей, обладающих общими характеристиками. В этом случае формируется обобщенная сущность (родовой предок), а специфическая для каждого типа информация может быть расположена в категориальных сущностях (потомках). Для каждой категории указывается дискриминатор – атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой.

Иерархии категорий делятся на два типа – полные и неполные. В полной категории одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке. 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.

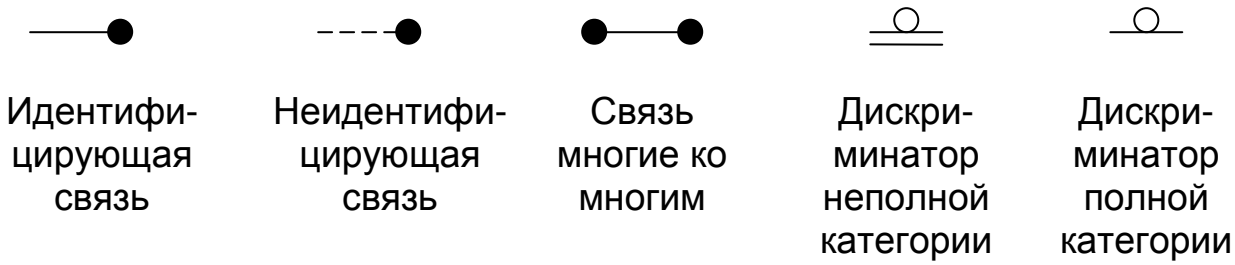


Рис. 12. Графические примитивы IDEF1X. Типы связей

Имя роли (функциональное имя) – это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности. Имя роли используется в случае, когда несколько атрибутов одной сущности имеют одинаковую область значений, но разный смысл или в случае рекурсивных связей (fish hook), когда одна и также сущность является и родительской и дочерней одновременно. При задании рекурсивной связи атрибут мигрирует в качестве внешнего ключа в состав неключевых атрибутов той же сущности. Так как атрибут не может появиться дважды в одной сущности под одним именем, он получает имя роли. Рекурсивная связь может быть только неидентифицирующей.

Описанные особенности проиллюстрированы на примере (см. рис. 13).

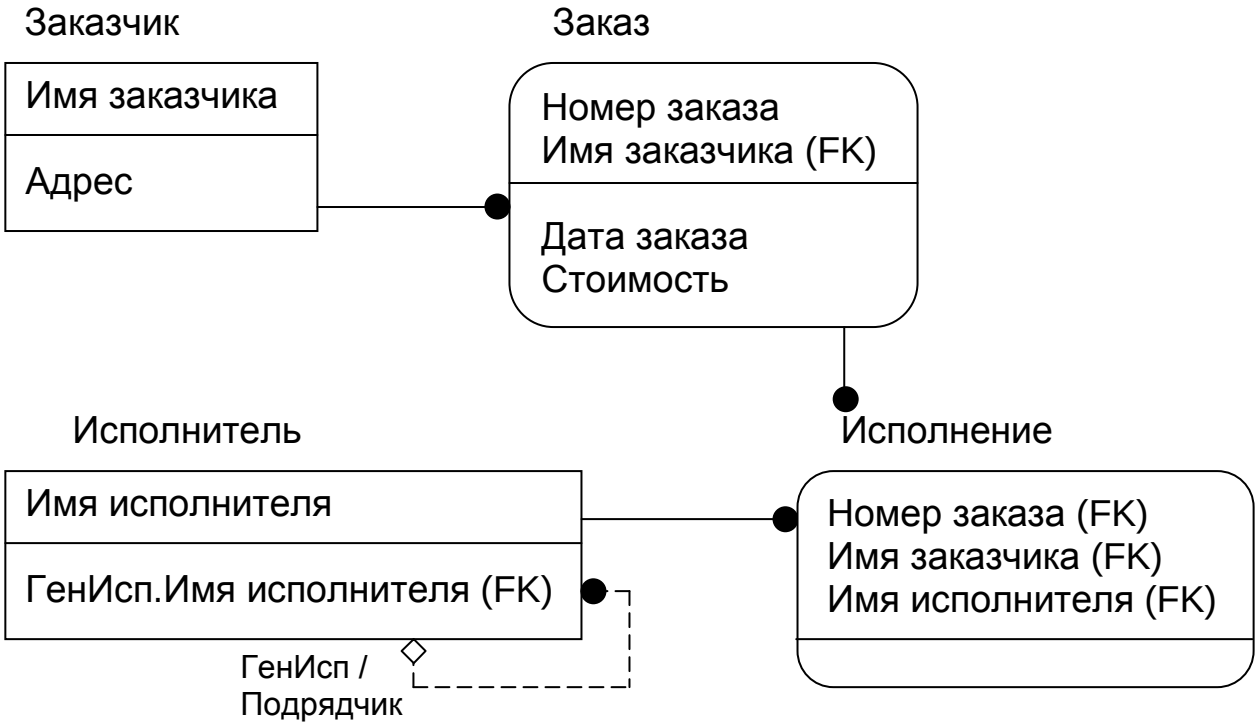


Рис. 13. ER модель базы данных по IDEF1X

5 Создание смешанной модели с использованием IDEF0, DFD, IDEF3 и IDEF1X

Для разностороннего описания автоматизированной системы обработки информации и управления возможно создание смешанной модели, содержащей диаграммы IDEF0, DFD, IFEF3 и IDEF1X. При этом возможны следующие переходы между нотациями:

- IDEF0 → DFD,
- IDEF0 → IDEF3,
- DFD → IDEF3.

Нельзя декомпозировать работу DFD с помощью диаграммы IDEF0 и работу IDEF3 с помощью диаграммы любой другой нотации.

Согласно нотации DFD диаграмма не должна иметь граничных стрелок – потоков данных. Поэтому при декомпозиции работы IDEF0 с помощью диаграммы DFD стрелки входа, выхода, управления и механизма на диаграмме декомпозиции DFD не отображаются. Вместо них необходимо создать соответствующие внешние сущности и хранилища данных и внутренние потоки данных между ними, а стрелки, входящие и выходящие из декомпозируемого блока на диаграмме IDEF0 поместить в тоннель.

При декомпозиции работы IDEF0 или DFD в IDEF3 стрелки не мигрируют на нижний уровень. Объекты, представленные стрелками на диаграмме IDEF0, могут быть отображены на декомпозирующей диаграмме IDEF3 в качестве объектов ссылки.

Модель данных IDEF1X строится на отдельной диаграмме с учетом функциональности, определенной диаграммами IDEF0, DFD и IDEF3.

При этом одной и той же сущности IDEF1X может соответствовать несколько объектов на диаграммах других нотаций и наоборот (например, одна стрелка IDEF0 может содержать данные, представленные атрибутами нескольких сущностей модели данных). Указанные соответствия определяются пояснительным текстом, дополняющим диаграммы смешанной модели.

Связывание стрелок IDEF0, потоков данных и хранилищ DFD и ссылок IDEF3 с сущностями IDEF1X позволяет обеспечить требуемую согласованность, корректность и завершенность смешанной модели.

Для построения смешанной модели существует следующий рекомендуемый порядок действий. Сначала необходимо определить цель моделирования, точку зрения на модель и описать область моделирования. Собственно моделирование начинают с построения контекстной диаграммы IDEF0 и диаграммы декомпозиции верхнего уровня IDEF0, на которой функциональные блоки декомпозируется с помощью диаграмм IDEF0, DFD и IDEF3. Дальнейшая декомпозиция продолжается до достижения цели моделирования. На основе и в соответствии с построенными диаграммами производится создание модели данных IDEF1X.

6 Объектно-ориентированное проектирование на языке UML

6.1 Общие сведения

Язык UML (Unified Modeling Language) представляет собой унифицированный язык визуального моделирования, который разработан для специфицирования (создания спецификации), конструирования, визуализации, и документирования компонентов программного обеспечения и бизнес-процессов. Язык UML [4 – 9] может быть использован для построения концептуальных и логических моделей сложных систем самого различного целевого назначения.

Конструктивное использование языка UML основывается на применении общих принципов объектно-ориентированного анализа и проектирования:

- принцип абстрагирования, который предписывает включать в модель только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций;
- принцип многомодельности, который представляет собой утверждение о том, что никакая единственная модель не может с достаточной степенью адекватности описывать различные аспекты сложной системы. При этом наиболее общими представлениями сложной системы принято считать статическое (структурное) и динамическое (описание логики процессов) представления;
- принцип иерархического построения моделей сложных систем, который предписывает рассматривать процесс построения модели на разных уровнях абстрагирования или детализации в рамках фиксированных представлений.

Процесс проектирования заключается в построении модели, записанной в графической нотации. При этом соблюдаются общие принципы структурного проектирования: нисходящая разработка, иерархическое построение модели, строгая формализация и четкая семантика.

Представления о модели сложной системы фиксируются на языке UML в виде специальных графических конструкций – диаграмм:

- вариантов использования (use case);
- классов (class);
- поведения (behavior):
  - состояний (state chart);
  - деятельности (activity);
  - взаимодействия (interaction):
    - последовательности (sequence);
    - кооперации (collaboration);
- реализации (implementation):
  - компонентов (component);
  - развертывания (deployment).

6.2 Диаграмма вариантов использования

Диаграмма вариантов использования описывает функциональное назначение системы. Проектируемая система представляется в виде множества сущностей или актеров, взаимодействие которых с системой отображается в виде взаимосвязанных вариантов использования.

Актером (Actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему. Вариант использования (Use case) служит для описания сервисов, которые система предоставляет актеру или прецедентов использования системы.

Взаимодействие экземпляров актеров и вариантов использования между собой описывается с помощью отношений:

- ассоциации – служит для обозначения специфической роли актера в отдельном варианте использования;
- включения – указывает, что некоторая последовательность поведения одного варианта использования включает в качестве составного компонента определенное поведение другого варианта использования (отношение определяется стрелкой от включающего к включаемому);
- расширения – отмечает тот факт, что один из вариантов использования может присоединить к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования (отношение определяется стрелкой от расширяющего к расширяемому);
- обобщения – применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями родительских вариантов и участвуют во всех отношениях родительских вариантов (отношение определяется стрелкой от потомка к предку). Дочерние варианты использования могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения;

Пример диаграммы вариантов использования описан на рис. 16. Вариант использования по приему заказа в обязательном порядке включает ввод параметров, поэтому используется отношение «include». Выбор специальной формы оплаты возникает только при необходимости, это расширяющий вариант использования и применяется отношение «extend». Прием VIP заказа – это специальный случай приема заказа, имеющего высокую важность.

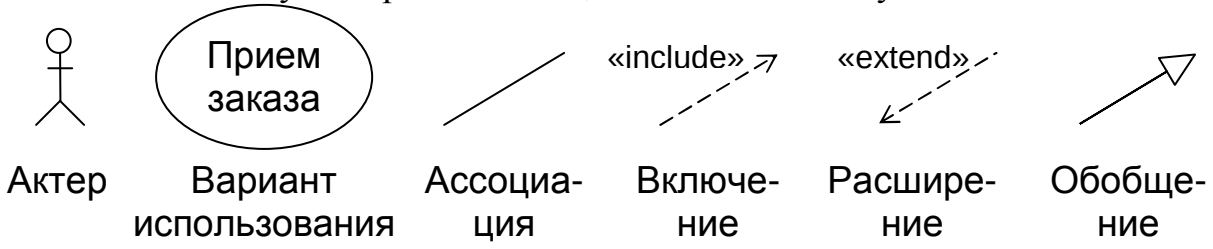


Рис. 14. Графические примитивы диаграммы вариантов использования UML

6.3 Диаграмма классов

Диаграмма классов служит для представления статической структурной модели системы в терминологии классов объектно-ориентированного проектирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, а также описывает их внутреннюю структуру и типы отношений. При этом на данной диаграмме не указывается информация о временных аспектах функционирования системы.

Класс (class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Описание класса состоит в определении атрибутов (свойств) и методов (операций или сервисов).

Интерфейс в языке UML – это семантическая и синтаксическая конструкция, используемая для специфицирования методов класса.

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений:

- ассоциации – соответствует наличию некоторого отношения между классами;
- агрегации – частный случай ассоциации, когда один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности;
- композиции – частный случай агрегации, при котором выделяется специальная форма отношения «часть-целое», при которой составляющие части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его части;
- обобщения – отношение между более общим элементом (родителем или предком) и более частным и специальным элементом (дочерним или потомком).
- зависимости – указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, выраженное в том, что некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели;

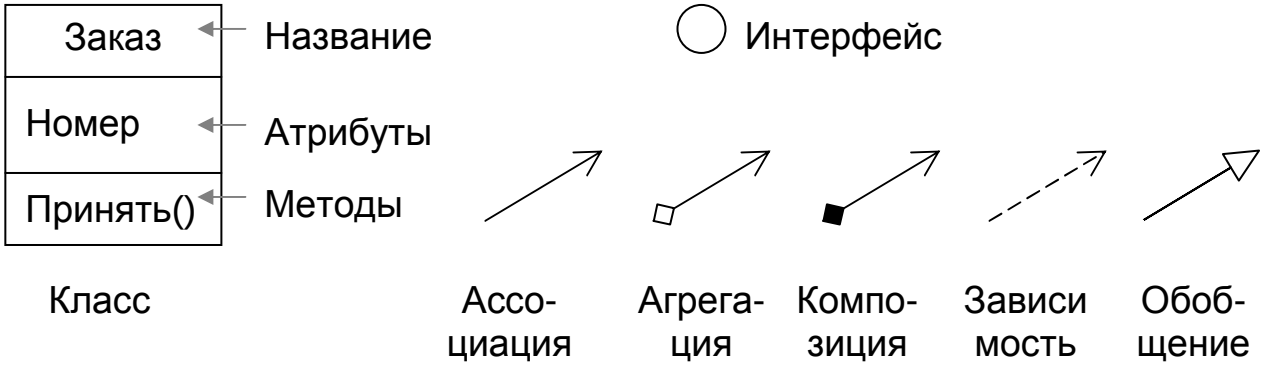


Рис. 15. Графические примитивы диаграммы классов UML

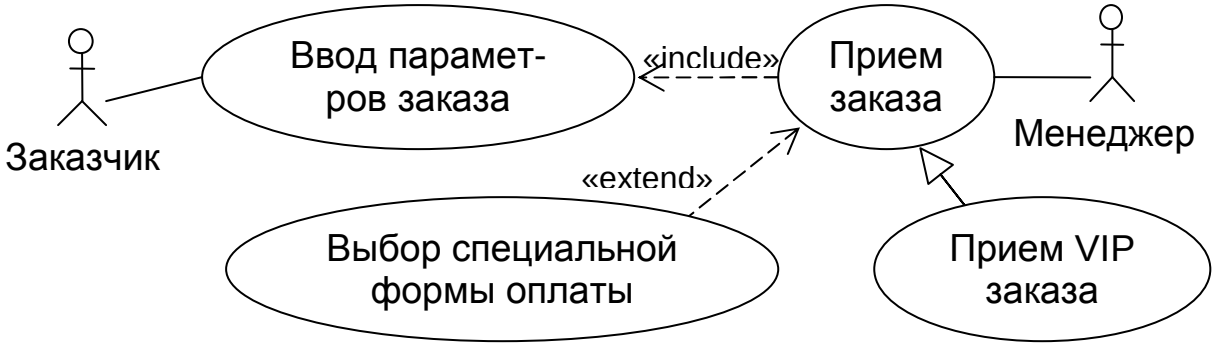


Рис. 16. Пример диаграммы вариантов использования UML

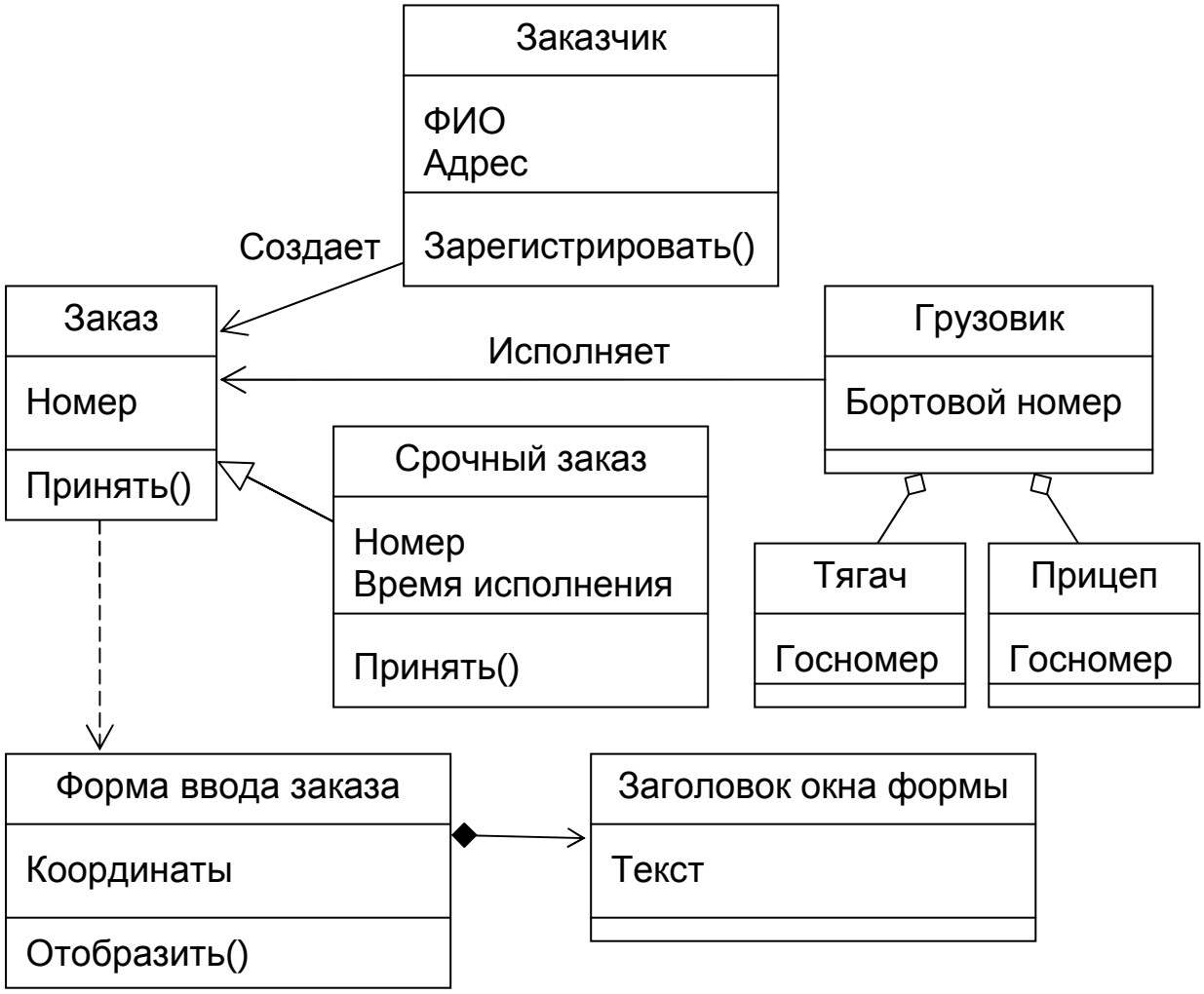


Рис. 17. Пример диаграммы классов UML

Рис. 17 иллюстрирует пример диаграммы классов. Заказчик задает заказ, а грузовик его исполняет, что описано с помощью соответствующих сущностных классов, связанных ассоциацией. Срочный заказ наследует все атрибуты обобщенного заказа и имеет свой атрибут – время исполнения. Форма ввода заказа позволяет вводить его параметры. Грузовик – это сущность, содержащая тягач и прицеп, которые, впрочем, могут выступать в системе и независимо. Заголовок окна формы существовать в отрыве от формы не может – поэтому используется композиция.

6.4 Диаграмма состояний

Диаграмма состояний описывает процесс изменения состояний одного или нескольких экземпляров классов, т. е. моделирует все возможные изменения в состоянии конкретного объекта, которые вызваны внешними воздействиями со стороны других объектов или извне. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Главное назначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, которые отражают динамический или функциональный аспект его поведения. При этом изменение их отдельных значений будет отражать изменение состояния. Состояние определяется именем и списком внутренних действий (деятельностей), которые выполняются в процессе нахождения моделируемого элемента в данном состоянии и характеризуются меткой действия (entry, exit, do, include).

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершины этого графа – состояния и некоторые другие типы элементов автомата (псевдосостояния), отображаемые соответствующими графическими символами.

Дуги графа служат для обозначения переходов из состояния в состояние. Срабатывание перехода зависит от наступления некоторого события и от выполнения определенного условия, называемого сторожевым. На базе простых переходов возможна организация сложных – соединение (две и более входящих дуг) и ветвление (две и более исходящих дуг). Диаграммы состояний могут быть вложены друг в друга, образуя составные состояния, которые могут быть последовательными, параллельными и историческими – т.е. запоминающими; синхронизирующими.

Существует два частных случая состояния. В начальном состоянии объект находится по умолчанию в начальный момент времени. В конечном состоянии объект находится после завершения работы автомата в конечный момент времени. Эти состояния не содержат никаких внутренних действий (псевдосостояний).

Рис. 18. Графические примитивы диаграммы состояний UML

6.5 Диаграмма деятельности

Диаграммы деятельности позволяют описать особенности процедурного и синхронного управления, обусловленного выполнением внутренних деятельностей и действий (элементарных операций). Диаграмма деятельности является специальным видом диаграммы состояний, на которой для отображения действий введены специальные состояния, а на переходах отсутствует сигнатура событий. При этом на этой диаграмме допускается использование и обычных состояний.

Основным направлением использования диаграмм деятельности является визуализация особенностей реализации методов классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции некоторого класса либо ее части.

Символы ветвления используется для описания условных переходов. Разделение (concurrent fork) и слияние (concurrent join) используются для разделения и слияния параллельных вычислений или потоков управления. Дорожки (swim lanes), изображенные на рис. 21 позволяют разделять выполнение процессов между классами или объектами (а также подразделениями в случае описания бизнес-процесса с помощью диаграммы деятельности).

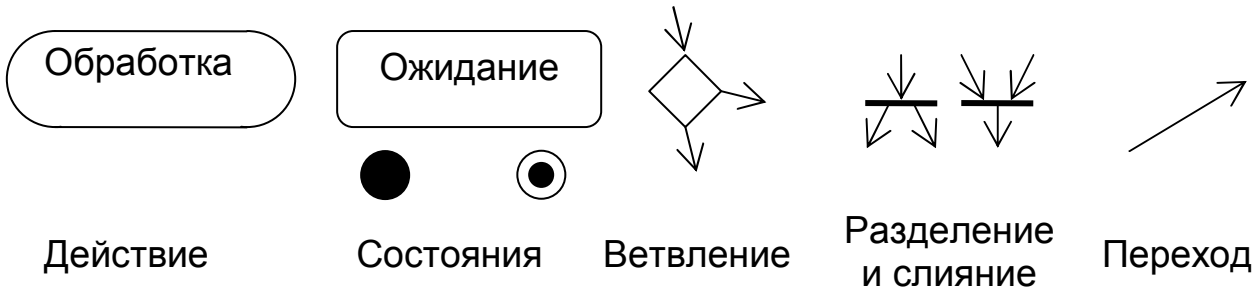


Рис. 19. Графические примитивы диаграммы деятельности UML

На рис. 20 приведен пример диаграммы состояний для описания изменения Заказа. В случае, когда заказ принят и может быть исполнен, он принимается к исполнению, прерывание которого возможно в случае отмены или успешного завершения.

На рис. 21 приведена диаграмма деятельности системы по приему заказа. Интерфейсный модуль позволяет вводить параметры заказа и отображать результат его исполнения. Система обработки заказа проводит проверку на выполнимость, а система планирования назначает исполнителя.

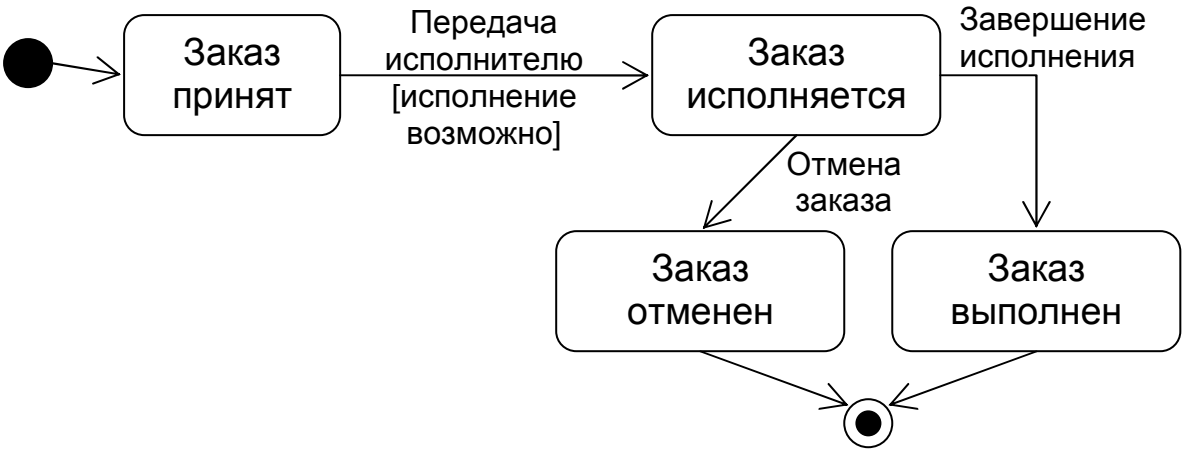


Рис. 20. Пример диаграммы состояний UML.

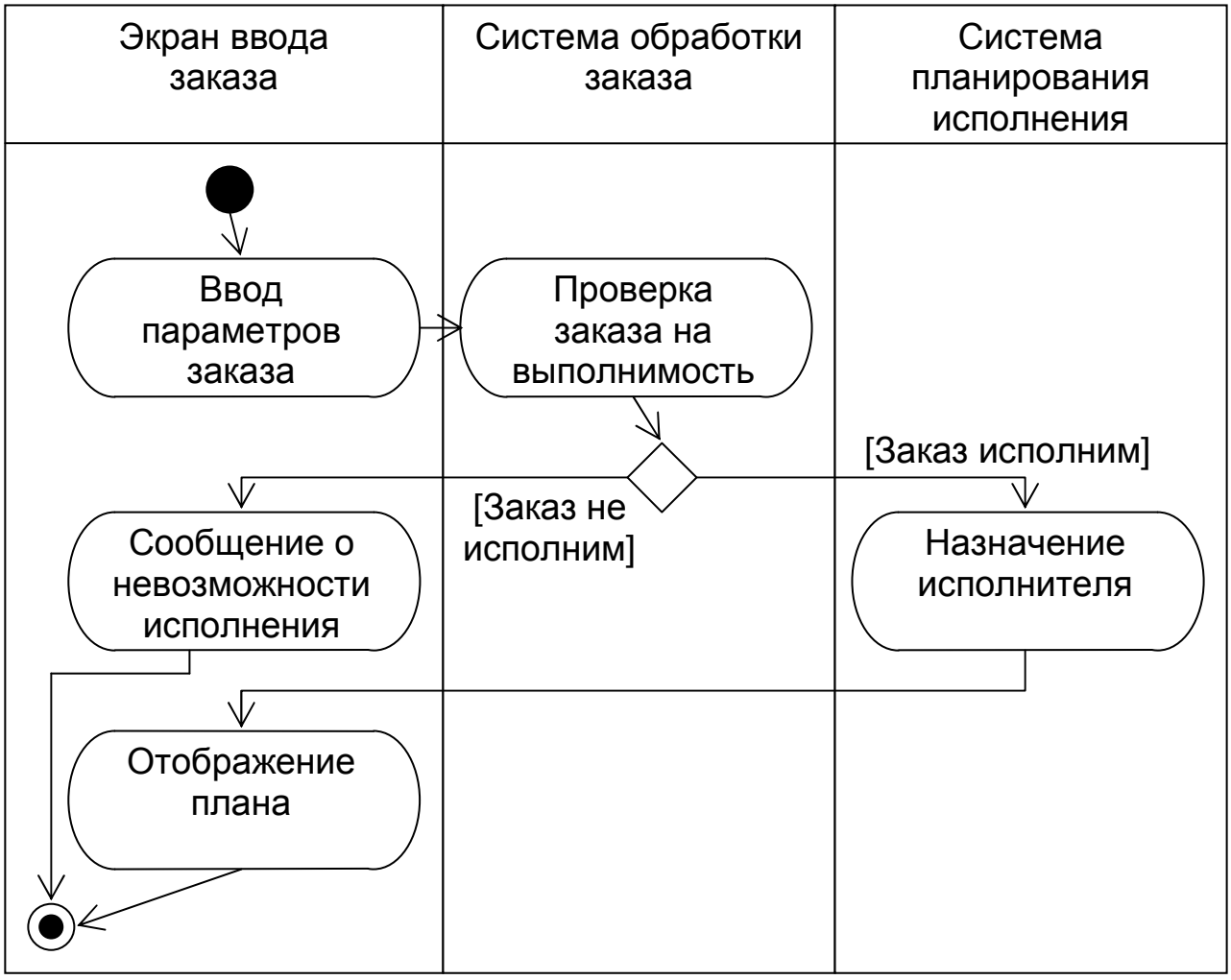


Рис. 21. Пример диаграммы деятельности UML

6.6 Диаграмма последовательности

Диаграмма последовательности отображает взаимодействие объектов – экземпляров классов во времени, выраженное в обмене сообщениями. При этом возможные статические ассоциации между объектами не показываются. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени (вертикальная временная ось направлена сверху вниз).

Линия жизни объекта, изображаемая в виде вертикальной линии, служит для обозначения периода времени, в течение которого объект активен, то есть участвует во взаимодействии.

Взаимодействия объектов реализуются посредством сообщений, которые образуют порядок по времени своего возникновения. Сообщение – законченный фрагмент информации, который отправляется одним объектом другому. При этом прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено.

В языке UML могут встречаться несколько разновидностей сообщений:

- вызов процедур, выполнение операций или обозначение отдельных вложенных потоков управления;
- обозначение простого (не вложенного) потока управления;
- асинхронное сообщение между двумя объектами в некоторой процедурной последовательности;
- возврат из вызова процедуры.

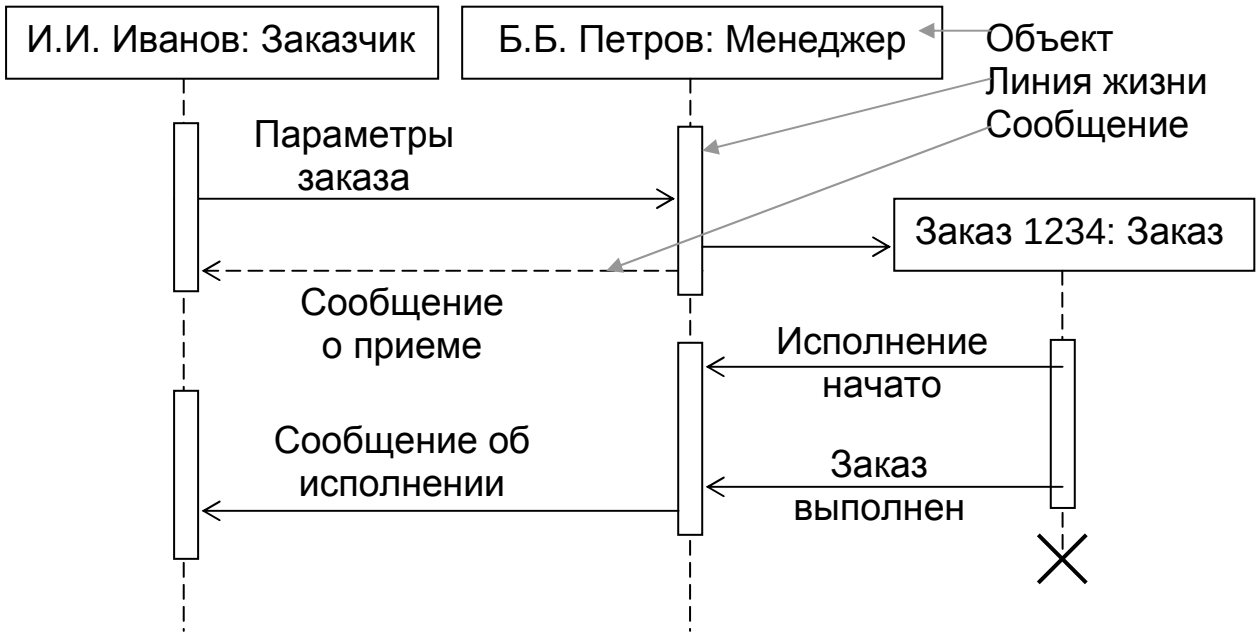


Рис. 22. Графические примитивы диаграммы последовательности UML и пример диаграммы

6.7 Диаграмма кооперации

Диаграмма кооперации предназначена для спецификации структурных аспектов взаимодействия. Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов, отображаются в виде прямоугольников. Ассоциации между объектами указываются в виде различных соединительных линий. При этом можно явно указать имена ассоциации и ролей, которые играют объекты в данной ассоциации. Дополнительно могут быть изображены динамические связи – потоки сообщений. Они представляются также в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

Сообщение на диаграмме кооперации специфицирует коммуникацию между двумя объектами, один из которых передает другому некоторую информацию. При этом первый объект ожидает, что после получения сообщения вторым объектом последует выполнение некоторого действия.

В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. С другой стороны, на этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров.

Кооперация может быть представлена на двух уровнях:

- на уровне спецификации – показывает роли классификаторов и роли ассоциаций в рассматриваемом взаимодействии;
- на уровне примеров – указывает экземпляры и связи, образующие отдельные роли в кооперации.

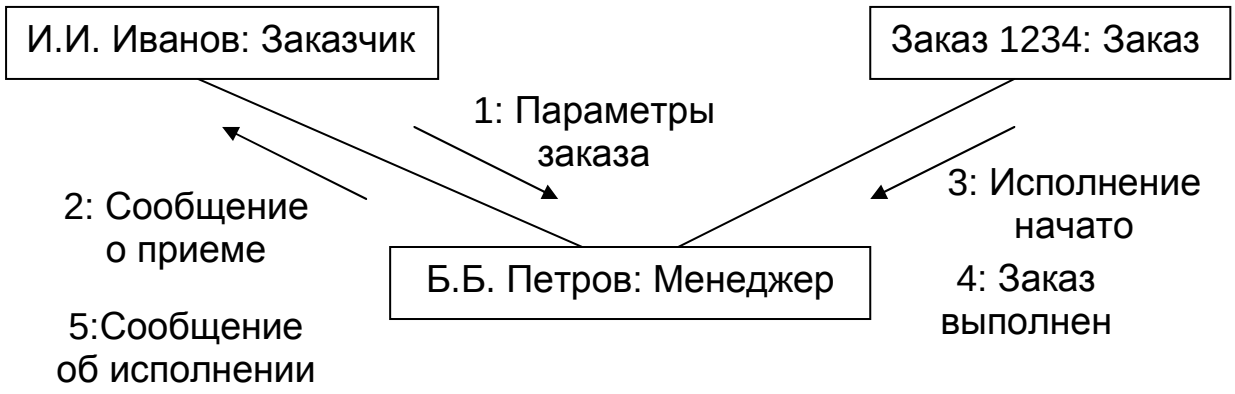


Рис. 23. Графические примитивы диаграммы кооперации UML и пример диаграммы

6.8 Диаграмма компонентов

Диаграмма компонентов описывает особенности физической реализации системы в момент перехода от логического представления к конкретной реализации системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

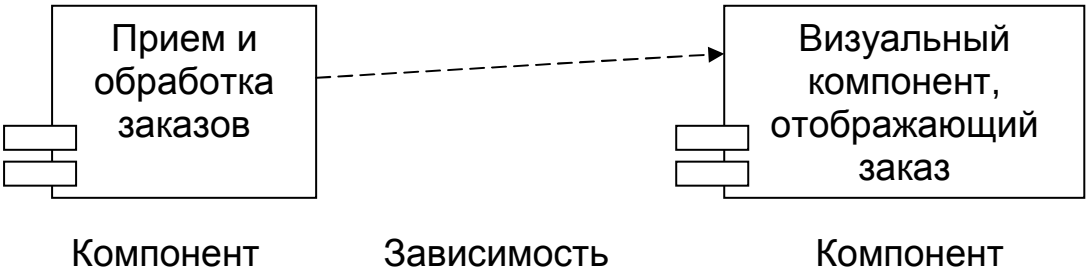


Рис. 24. Графические примитивы диаграммы компонентов UML и пример диаграммы

6.9 Диаграмма развертывания

Диаграмма развертывания (размещения) применяется для представления общей конфигурации и топологии распределенной автоматизированной системы и содержит распределение компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы. Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними.

На рис. 25 представлен классический пример трехзвенной архитектуры автоматизированной системы в виде диаграммы развертывания UML.

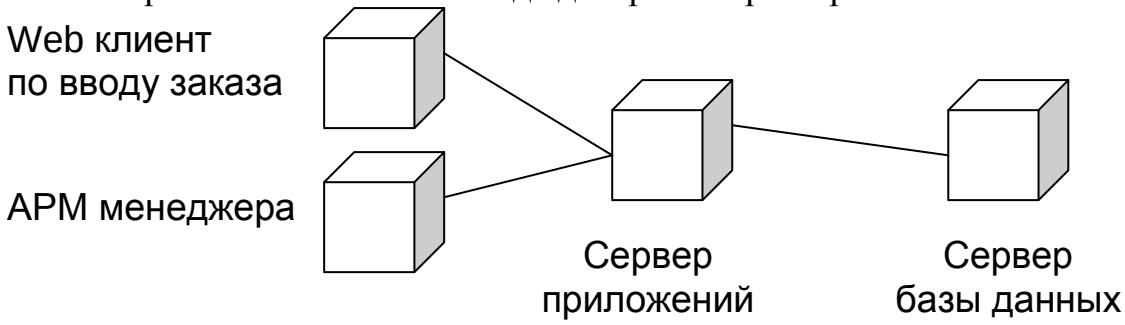


Рис. 25. Графические примитивы диаграммы развертывания UML и пример диаграммы

7 Процессный подход к проектированию ARIS

7.1 Общие сведения

Методология ARIS (Architecture of Integrated Information Systems – архитектуры интегрированных информационных систем) состоит в описании модели процессов [10] в виде четырех интегрированных между собой представлений:

- функциональное представление содержит описание функций бизнес-процесса или системы, отдельных подфункций (операций) и их взаимосвязи между собой;
- информационное представление описывает состояния информационных объектов (данных), и события, приводящие к их изменению;
- организационное представление определяет совокупность организационных единиц и их взаимосвязей;
- управляющее представление описывает взаимосвязи между указанными представлениями.

Каждое представление содержит разные диаграммы (ARIS поддерживает разнообразные графические нотации), которые по времени их возникновения относят к трем последовательным уровням или этапам проработки представления (см. Таблицу 4):

- на уровне описания требований происходит определение целей моделирования, языка предметной области и программного решения рассматриваемой задачи, которое базируется на результатах анализа проблем бизнеса и позволяет описать формализованные требования к системе;
- на уровне спецификации проекта концептуальные понятия, сформулированные на предыдущем уровне формулировки требований, трансформируются в категории, методы и алгоритмы в терминах информационных технологий;
- на уровне описания реализации спецификация проекта трансформируется в конкретные аппаратные и программные компоненты.

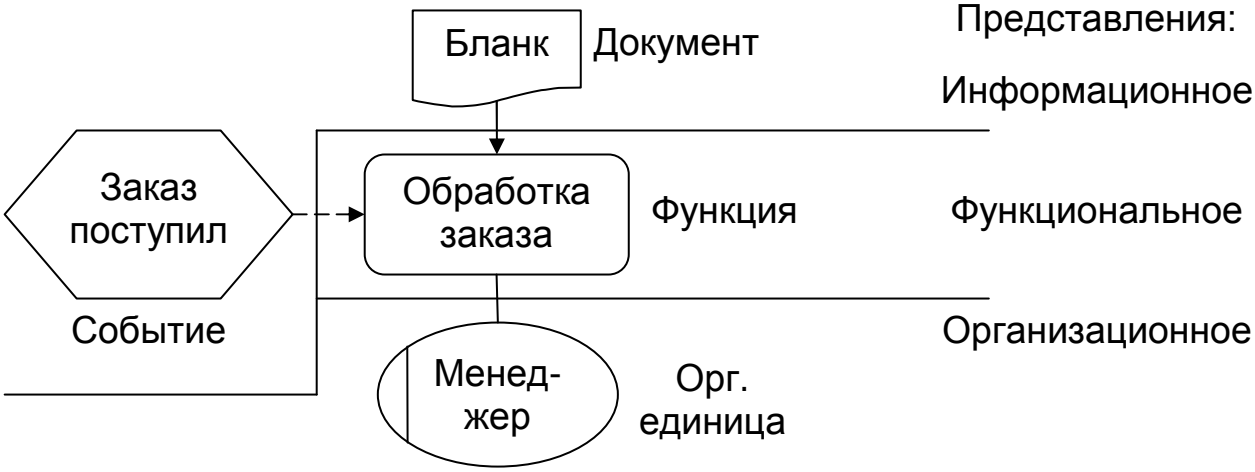


Рис. 26. Представления ARIS и базовые графические примитивы

Таблица 4. Уровни представлений ARIS

Уровни проработки	**Представления**		
	Функциональное	**Информационное**	**Организационное**
Описание требований	Иерархия функций в виде дерева	ER модель в нотации Чена (Сущности, Связи и Атрибуты)	Иерархия организационных единиц в виде организационной схемы (Должности, Типы сотрудников, Сотрудники)
Спецификация	Иерархия типов прикладных систем и типов модулей, реализующих функции	Реляционная диаграмма и диаграмма атрибутов (Отношения и Атрибуты)	Топология сети (Узлы и компоненты оборудования)
Реализация	Диаграмма прикладной системы, отражающей фактическую реализацию	Табличная диаграмма (Таблицы и Поля)	Диаграмма сети с учетом конкретного местоположения

Управляющее представление содержит нотации VAD и eEPC/PCD, которые описаны ниже.

Моделирование АСОИУ по методологии ARIS может содержать следующие этапы:

1. Определение цели моделирования и точки зрения на проект.
2. Построение диаграммы VAD и определение управляющей модели на высоком уровне.
3. Построение диаграмм eEPC/PCD, описывающих логику бизнес-процессов в виде взаимодействующих событий, функций, информационных объектов и организационных единиц.
4. Построение иерархий функций, данных и организационных единиц на уровне описания требований путем обобщения или уточнения абстракций, появившихся на диаграммах VAD и eEPC/PCD.
5. Проработка функционального, информационного и организационного представлений на уровнях спецификации и реализации.
6. Корректировка eEPC/PCD и VAD диаграмм по результатам проработки функционального, информационного и организационного представлений.

7.2 Нотация VAD

Диаграммы VAD (Value Added Chain Diagrams) описывают цепочки процессов, добавляющих стоимость, и используются для представления процесса на верхнем уровне. Процессы, или некоторые группы функций соединяются между собой пунктирной стрелкой, которая имеет тип «is predecessor of» (является предшественником). При этом возможно существование обратной связи.

Между процессами отображаются потоки материальных ресурсов и информации. Для описания инфраструктуры, необходимой для выполнения процесса, используются организационные единицы и ресурсы (типа «Product Service» и «Information Service»). Связь с процессами обозначается сплошной линией, при этом может быть указан тип связи.

Пример диаграммы VAD для процесса обработки заказа представлен на рис. 28. Указанные процессы добавляют стоимость в смысле обеспечения прибыльности. Между процессами передаются информационные и материальные объекты.

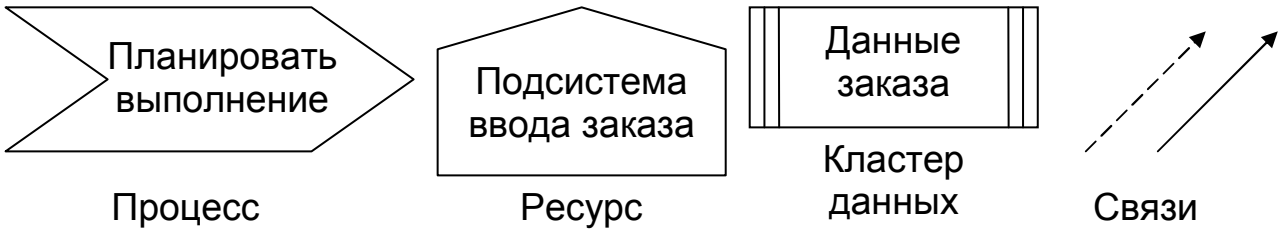


Рис. 27. Специфичные графические примитивы VAD диаграмм

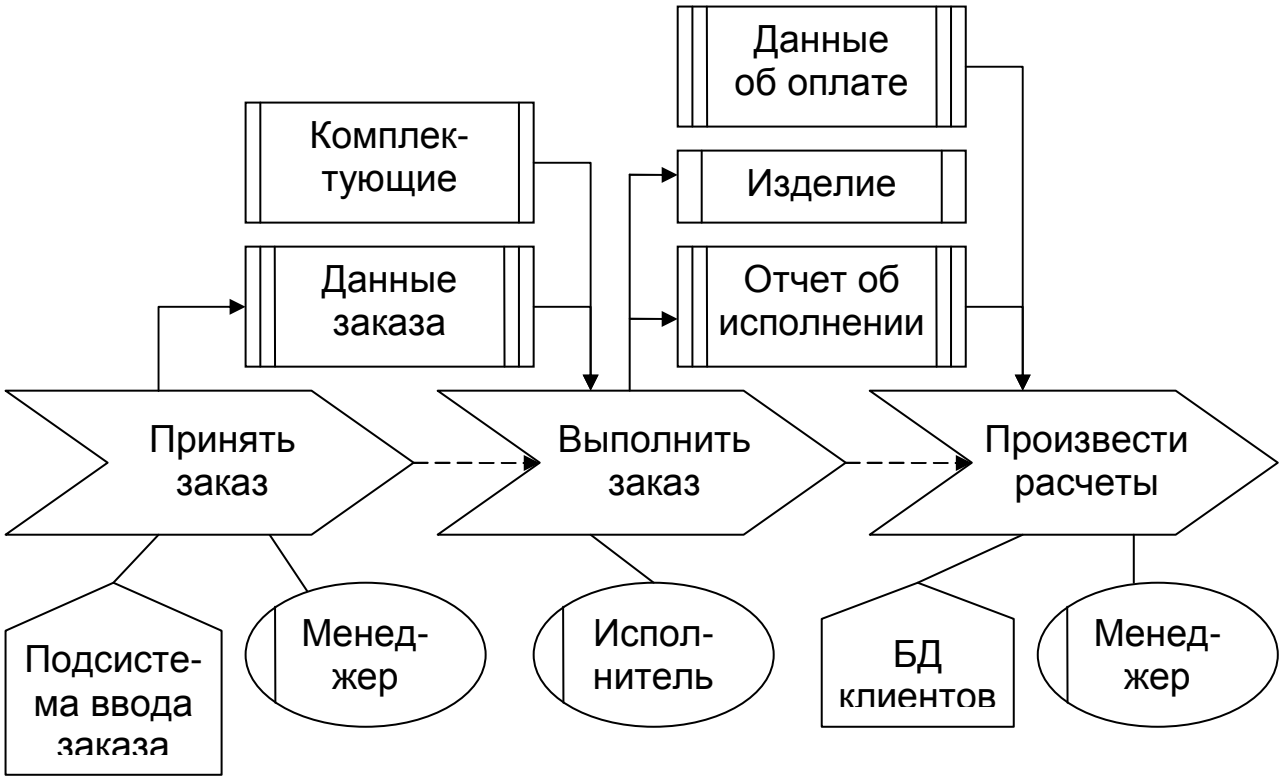


Рис. 28. Пример диаграммы VAD

7.3 Нотация eEPC и PCD

Расширенные цепочки процессов, управляемых событиями (eEPC – Extended Event Driven Process Chains) предназначены для описания сложных процессов и содержат события, функции, информационные объекты и организационные единицы (см. рис. 29 – 30).

Событие служит для отображения возможных результатов выполнения функций и инициируют выполнение новых функций. Каждая функция должна быть инициирована событием и завершена событием. В каждую функцию не может входить более одной стрелки, инициирующей ее выполнение, и выходить не более одной стрелки, описывающей завершение ее выполнения.

Для описания ветвления используются операторы. Возможны разные соединения операторов и функций кроме двух: после события нельзя использовать операторы ИЛИ и исключающего ИЛИ, так как решение может приниматься только в процессе выполнения функции.

Для соединения событий и функций используются пунктирные линии, для соединения функций, информационных объектов и организационных единиц – сплошные.

Графические элементы на диаграмме eEPC могут располагаться в произвольном порядке, однако обычно выбирают направление слева направо или сверху вниз.

Диаграммы цепочки процессов (PCD – Process Chain Diagrams) являются разновидностью диаграмм eEPC. На этих диаграммах графические элементы распределяются по столбцам, что в некоторых случаях облегчает их читаемость. Такое представление лучше подходит при документировании, но неудобно при описании процессов с разветвленной логикой.

Пример диаграммы PCD для процесса обработки заказа приведен на рис. 31. Первый и второй столбцы диаграммы PCD отображают логическую последовательность выполнения процесса – они содержат события и функции, связанные между собой пунктирными стрелками и операторами.

Количество столбцов может быть расширено. Возможно добавление таких столбцов, как «Носитель», «Прикладная система» и «Ресурс» (заполняемых соответствующими объектами) и «Экран», «Пакет», «Диалог» (заполняемых аналогично столбцу «Руководство»).

Диаграммы процесса могут быть использованы при анализе реальных бизнес-процессов, определении недостатков их организации или причин неэффективности. Такими недостатками могут быть дезинтеграция между ручной обработкой и обработкой с помощью информационных технологий или организационная дезинтеграция. Кроме того, проявляются лишние входы (процедурная избыточность данных) и задержки в выполнении.

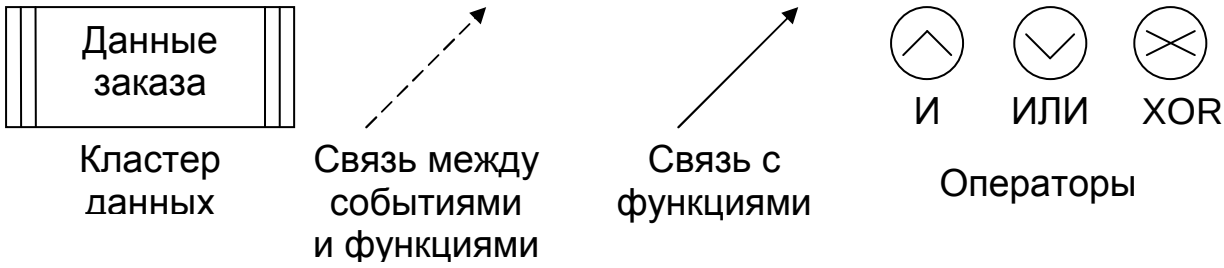


Рис. 29. Специфичные графические примитивы eEPC и PCD диаграмм

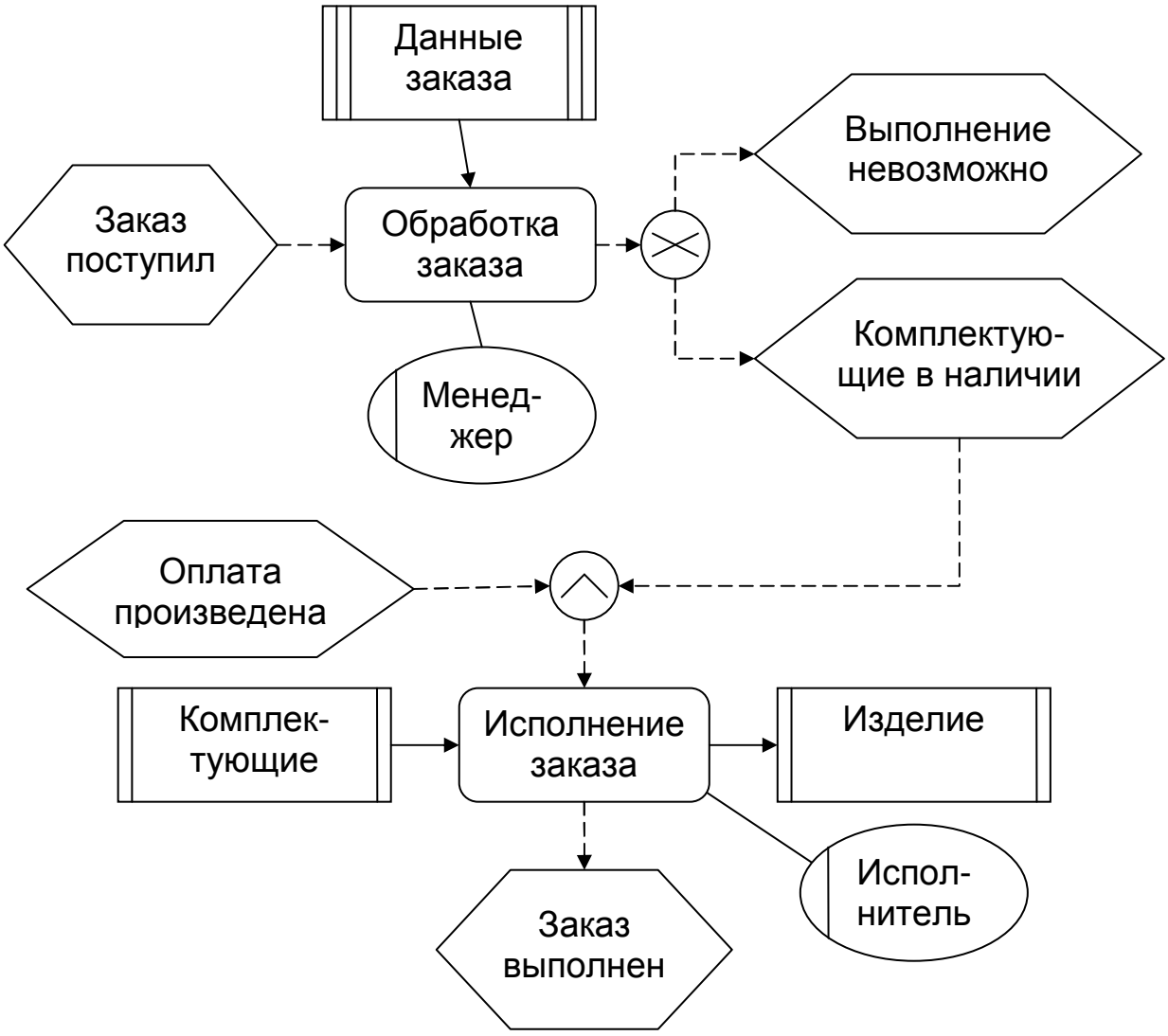


Рис. 30. Пример eEPC диаграммы

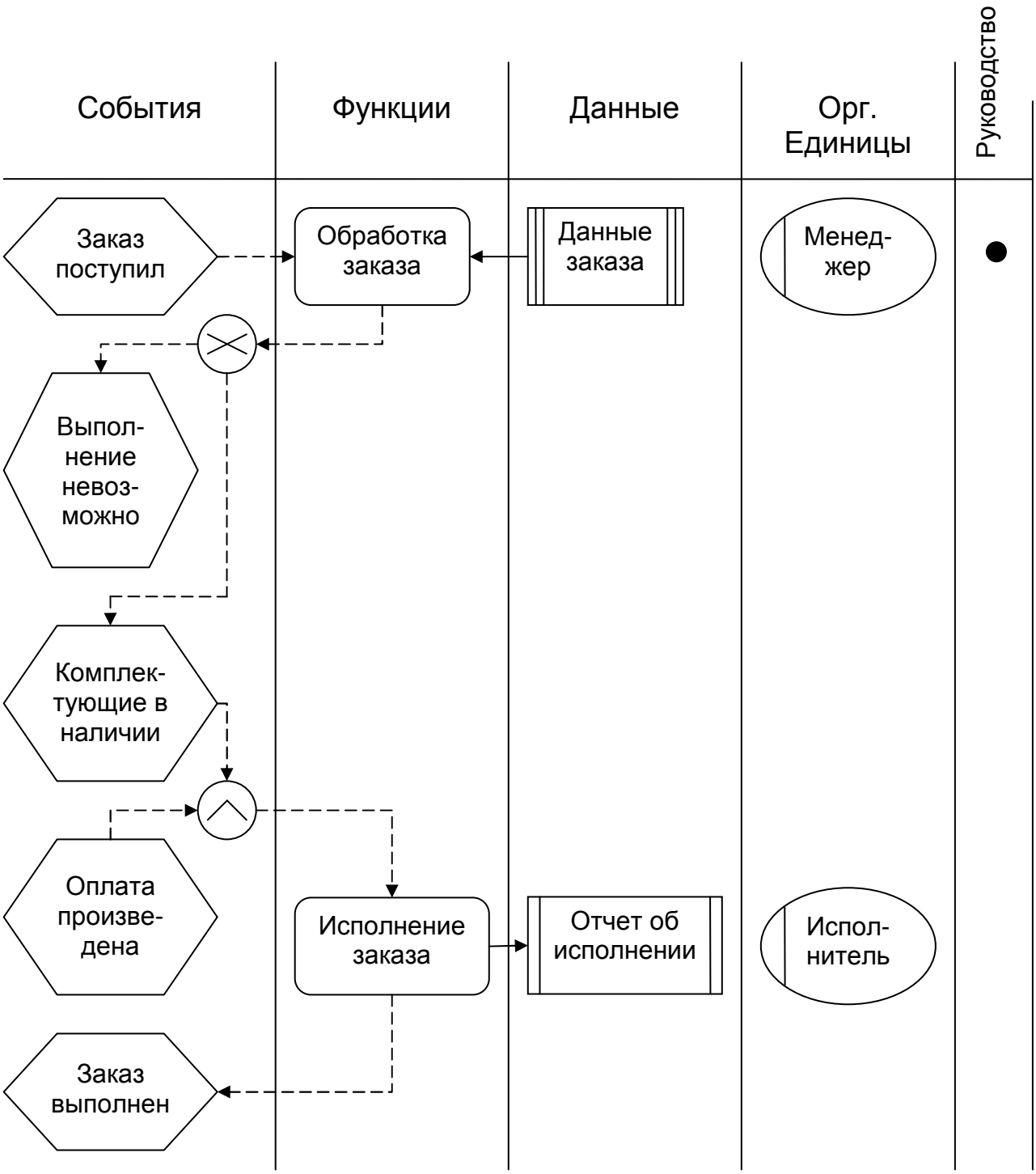


Рис. 31. Пример PCD диаграммы

8 Заключение

Рассмотренные методологии проектирования АСОИУ обладают рядом общих свойств:

1. Проектирование начинается с определения цели и точки зрения.
2. Модель представляется в виде иерархии упорядоченных диаграмм, описывающих разные аспекты модели.
3. При этом происходит описание функций (IDEF0 и DFD, диаграммы вариантов использования, последовательности и кооперации UML, VAD диаграмма и функциональное представление ARIS), процессов (IDEF3, диаграммы деятельности и состояний UML, eEPC/PCD диаграммы ARIS) и статической структуры (IDEF1X, диаграммы классов UML, информационное и организационное представление ARIS).
4. При построении диаграмм необходимо строго придерживаться выбранной графической нотации, согласно которой абстракции одного класса имеют одинаковое графическое отображение.
5. Обязательным компонентом модели является краткое текстовое описание диаграмм.

Также при применении методологий проектирования АСОИУ необходимо руководствоваться общим принципом:

1. Сначала сформулировать идею, которая должна быть отражена на диаграмме (в виде фразы на русском (или любом другом) языке без ошибок).
2. Только затем отобразить основное содержание этой фразы на диаграмме с использованием графической нотации.

Следует избегать формального подхода к проектированию, когда в модель включаются все необходимые составляющие только для того, чтобы соблюсти процедуру проектирования, но не для того, чтобы описать важные идеи. Диаграммы или их фрагменты, которые не содержат существенной семантической нагрузки, должны исключаться из модели.

В целом, необходимо соблюдать два основных требования:

- информационно-логическая модель должна быть адекватной и достаточно детальной;
- описание модели должно быть ясным и наглядным, одновременно понятным заказчику и исполнителю.

Использование современных методологий проектирования и CASE-технологий позволяет выполнить эти требования, а грамотная организация процесса разработки обеспечивает высокую эффективность применения результатов проектирования.

Литература

1. Маклаков С.В. Моделирование бизнес-процессов с AllFusion Process Modeller (Bpwin 4.1). – М.: Диалог-МИФИ, 2003. – 240 с.

2. Маклаков С.В. Создание информационных систем с AllFusion Modelling Suite. – М.: Диалог-МИФИ, 2003. – 432 с.

3. Марка Д.А., МакГоуэн К. Методология структурного анализа и проектирования. – М.: МетаТехнология, 1993. – 111 с.

4. Буч Г, Рамбо Дж., Джекобсон А. UML. Проектирование программных комплексов, информационных систем. – М.: ДМК Пресс, СПб.: Питер, 2003, – 432 с.

5. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. – М.: ДМК Пресс; СПб.: Питер, 2004. – 432 с.

6. Гома Хасан. UML. Проектирование систем реального времени, параллельных и распределенных приложений: Пер. с англ. – М.: ДМК Пресс, 2002. – 704 с.

7. Леоненков А.В. Самоучитель UML. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2004. – 432с.

8. Рамбо Дж., Якобсон А., Буч Г. UML: специальный справочник. – СПб.: Питер, 2002. – 656 с.: ил.

9. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. – СПб: Питер, 2007. – 544 с.: ил.

10.Репин В.В., Елиферов В.Г. Процессный подход к управлению. Моделирование бизнес-процессов – 5-е изд. – М.: РИА «Стандарты и качество», 2007. – 408 с., ил.

Составитель:
Антон Владимирович Иващенко

Основы методологий проектирования
автоматизированных систем
обработки информации и управления

Издательство Самарского научного центра РАН
Лицензия на издательскую деятельность
ЛР № 040910 от 10.08.98 г.

Подписано в печать 07.04.09
Формат 60x84 1/8 Бумага офсетная. Печать офсетная.
Гарнитура Times New Roman. Усл. печ. л. 2,5
Тираж 300 экз. Заказ № 265

Отпечатано в типографии АНО «Издательство СНЦ РАН»
443001, г. Самара, Студенческий переулок, 3а
тел.: 242-37-07

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)

Методические рекомендации к проведению практических занятий по курсу «Современные проблемы информатики и вычислительной техники»

Составитель: к.т.н., доцент кафедры ИСТ
Куликовских И.М.

Самара 2013 г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ

Задание 1. Постановка задачи создания системы регистрации для учебного заведения

Перед руководителем информационной службы университета ставится задача разработки новой клинт-серверной системы регистрации студентов взамен старой системы на мейнфрейме. Новая система должна позволять студентам регистрироваться на курсы и просматривать свои табели успеваемости с персональных компьютеров, подключенных к локальной сети университета. Профессора должны иметь доступ к онлайновой системе, чтобы указать курсы, которые они будут читать, и проставить оценки за курсы.

Из-за недостатка средств университет не в состоянии заменить сразу всю существующую систему. База данных, содержащая всю информацию о курсах (каталог курсов), остается функционировать в прежнем виде. Эта база данных поддерживается реляционной СУБД. Новая система будет работать с существующей БД в режиме доступа, без обновления.

В начале каждого семестра студенты могут запросить каталог курсов, содержащий список курсов, предлагаемых в данном семестре. Информация о каждом курсе должна включать имя профессора, наименование кафедры и требования к предварительному уровню подготовки (прослушанным курсам).

Новая система должна позволять студентам выбирать 4 курса в предстоящем семестре. В дополнение каждый студент может указать 2 альтернативных курса на тот случай какой-либо из выбранных им курсов окажется уже заполненным или отмененным. На каждый курс может записаться не более 10 и не менее 3 студентов (если менее 3, то курс будет отменен). В каждом семестре существует период, когда студенты могут изменить свои планы. В это время студенты должны иметь доступ к системе, чтобы добавить или удалить выбранные курсы. После того как процесс

регистрации некоторого студента завершен, система регистрации направляет информацию в расчетную систему, чтобы студент мог внести плату за семестр. Если курс окажется заполненным в процессе регистрации, студент должен быть извещен об этом до окончательного формирования его личного учебного плана.

В конце семестра студенты должны иметь доступ к системе для просмотра своих электронных табелей успеваемости. Поскольку эта информация конфиденциальная, система должна обеспечить ее защиту от несанкционированного доступа.

Профессора должны иметь доступ к онлайновой системе, чтобы указать курсы, которые они будут читать, и просмотреть список студентов, записавшихся на их курсы. Кроме того, профессора должны иметь возможность проставить оценки за курсы.

Задание 2. Построить диаграмму вариантов использования системы обработки заказов

Компания – торговый посредник, продающая товары различных производителей, разрабатывает систему обработки заказов.

Дважды в год компания публикует каталог продуктов, который рассылается клиентам и другим заинтересованным лицам.

Клиенты приобретают товары, направляя в компанию перечень продуктов с информацией об оплате. Компания выполняет заказы и отправляет товары по адресам клиентов.

Система должна отслеживать заказ от момента его получения до отправки товара.

Клиенты могут возвращать товары, оплачивая, возможно, при этом некоторые издержки.

Часть клиентов заказывает товары через Интернет.

компания пользуется услугами различных транспортных и страховых компаний.

Задание 3. Построить диаграмму вариантов использования для системы кредитования коммерческого банка

Небольшой банк автоматизирует деятельность, связанную с кредитованием физических и юридических лиц (индивидуальных клиентов и организаций).

В настоящее время кандидат на получение кредита заполняет бумажную форму, прикладывает необходимые документы (финансовый отчет, перспективную оценку финансового состояния и др.) и отправляет в банк. Референт по кредитованию анализирует запрос на предмет возможных ошибок и подтверждает его достоверность.

Затем референт запрашивает отчет о кредитных операциях клиента в отделе кредитования. Копия отчета просматривается банковским служащим, а референт проверяет финансовое положение и доход клиента. Служащий также обращается к существующей системе управления счетами клиентов, чтобы получить необходимую информацию о состоянии счета и предыдущих кредитах клиента.

Вся информация комплектуется в кредитный запрос и направляется для оценки инспектору по кредитам. Если запрос утверждается, инспектор определяет наилучшие условия кредитования и уведомляет об этом клиента. Если клиент принимает условия, то кредит оформляется.

На обработку запроса обычно уходит минимум две недели (как для индивидуальных клиентов, так и для организаций).

Цель автоматизации – сократить время обработки запроса до 48 часов для индивидуальных клиентов и 72 часов для организаций, сократить количество сотрудников, занятых в процессе обработки, и увеличить количество запросов, обрабатываемых в заданный период.

Задание 4. Выполнить учебный проект для системы начисления зарплаты

Перед информационной службой компании поставлены задача создания новой системы начисления зарплаты взамен морально устаревшей существующей системы. Новая система должна предоставлять служащим возможность записывать электронным способом информацию из карточки учета рабочего времени и автоматически формировать чеки на оплату, учитывающие количество отработанных часов и общий объем продаж (для служащих, получающих комиссионное вознаграждение).

Новая система должна предоставлять служащим возможность вводить информацию из карточки учета рабочего времени, вводить заказы на поставку, изменять свои параметры (такие, как способ оплаты за работу) и формировать различные отчеты. Система должна работать на персональных компьютерах служащих всей компании. В целях обеспечения безопасности и аудита служащие должны иметь возможность доступа и редактирования только своих собственных карточек учета рабочего времени и заказов на поставку.

В системе должна храниться информация обо всех служащих компании в различных странах. Система должна обеспечивать правильную и своевременную оплату работы каждого служащего в соответствии с указанным им способом. Компания из соображений экономии желает сохранить без изменений одну из существующих баз данных, которая содержит всю информацию относительно проектов и тарифов. БД управления проектами функционирует в среде DB2 на мейнфрейме IBM. Новая система может читать данные из БД управления проектами, но не может обновлять их.

Часть служащих получает почасовую оплату. Она начисляется на основе карточек учета рабочего времени, каждая из которых содержит дату и

количество часов, отработанных в соответствии с конкретным тарифом. Если какой-либо служащий отработал в день более 8 часов, сверхурочное время оплачивается с коэффициентом 1,5. Служащие-почасовики получают зарплату каждую пятницу.

Некоторые служащие получают фиксированный оклад, однако они тоже представляют свои карточки учета рабочего времени. Благодаря этому система может вести учет количества часов, отработанных в соответствии с конкретными тарифами. Такие служащие получают зарплату в последний рабочий день месяца.

Некоторые из служащих с фиксированным окладом также получают комиссионное вознаграждение, учитывающее объем продаж. Они представляют заказы на поставку, отражающие дату и объем продаж. Процент комиссионного вознаграждения определятся индивидуально для каждого служащего и может составлять 10, 15, 25 или 35%.

Одной из наиболее часто используемых возможностей новой системы является формирование различных отчетов: запросить количество отработанных часов, суммарную зарплату, оставшееся время отпуска и т.д.

Служащие вправе выбирать способ оплаты за работу. Они могут получать свои чеки на оплату по почте, на счет в банке или на руки в офисе.

Администратор системы курирует информацию о служащих. В его обязанности входит ввод данных о новых служащих, удаление данных и изменение любой информации о служащем, такой, как имя, адрес и способ оплаты, а также формирование различных отчетов для руководства.

Приложение Начисление зарплаты запускается автоматически каждую пятницу и в последний рабочий день месяца, рассчитывая в эти дни зарплату соответствующих служащих. Начисление зарплаты должно производиться автоматически, без ручной обработки.

Система не должна позволять служащим изменять любые карточки учета рабочего времени, кроме своих собственных. Только администратор

системы может изменять любую информацию о служащих, за исключением способа доставки оплаты.

Система должна взаимодействовать с существующей банковской системой посредством интерфейса электронных трансакций.

Система должна обеспечивать Windows-совместный пользовательский интерфейс.

Задание 5. Выполнить учебный проект для информационной системы колледжа

В колледже работают N преподавателей. О каждом преподавателе известна следующая информация: фамилия, имя, отчество, год рождения, пол, образование, учебное заведение, которое он окончил (предполагается, что каждый из них окончил не более одного специального учебного заведения, если он окончил среднее и высшее учебное заведение, то фиксируется информация только о последнем из них), специальность, иностранные языки, которыми владеет преподаватель, и степень владения ими, адрес, информация о детях (ФИО, год рождения).

Каждый преподаватель может вести один или несколько предметов. За каждой группой студентов закреплен один руководитель.

Имеется расписание занятий, в котором зафиксировано, в какое время, в какой день недели, в какой аудитории, какая группа занимается, каким предметом и какой преподаватель его ведет.

Некоторые преподаватели ведут специальные семинары. Каждый преподаватель может вести несколько семинаров. Имеется расписание работы семинаров. Известно, кто из студентов посещает каждый семинар.

В колледже работают спортивные секции. Занятия в каждой из секций ведут несколько тренеров. Занятия в секциях ведутся по группам. Каждый студент может заниматься в одной или нескольких секциях. В каждой из этих

секций он записан в определенную группу. Каждая группа закреплена за одним определенным тренером.

Имеется расписание работы секций, в котором указано, какая группа, в какое время, с каким тренером занимается, а также место проведения занятий. Каждый тренер для каждой группы ведет журнал посещения занятий ее членами.

Задание 6. Выполнение учебного проекта.

Администрация больницы заказала разработку информационной системы для отдела приема пациентов и медицинского секретариата. Новая система предназначена для обработки данных о врачах, пациентах, приеме пациентов и лечении. Система должна выдавать отчеты по запросу врачей или администрации. Во время предпроектного обследования составлено следующее описание деятельности рассматриваемых подразделений.

Перед приемом в больницу проводится встреча пациента и врача. Врач сообщает в отдел приема пациентов об ожидаемом приеме больного и передает данные о нем. Пациент может быть принят в больницу более чем один раз, но если пациент ранее не лечился в больнице, то ему присваивается регистрационный номер и записываются его данные (фамилия, имя и отчество, адрес и дата рождения). Пациент должен быть зарегистрирован в системе до приема в больницу.

Спустя некоторое время врач оформляет в отделе приема пациентов прием больного. При этом определяется порядковый номер приема и запоминаются данные приема пациента. После этого отдел приема посылает сообщение врачу для подтверждения приема больного. В это сообщение включается регистрационный номер пациента и его фамилия, порядковый номер приема, дата начала лечения и номер палаты.

В день приема пациент сообщает в отдел приема о своем прибытии и передает данные о себе (или изменения в данных). Отдел приема проверяет и

при необходимости корректирует данные о пациенте. Если пациент не помнит свой регистрационный номер, то выполняется соответствующий запрос. После регистрации пациент получает регистрационную карту, содержащую ФИО пациента, адрес, дату рождения, номер телефона, группу крови, название страховой компании и номер страховки.

Во время пребывания в больнице пациент может лечиться у нескольких врачей; каждый врач назначает один или более курсов лечения, но каждый курс лечения назначается только одним врачом. Данные о курсах лечения передаются в медицинский секретариат, который занимается координацией лечения пациентов, регистрируются и хранятся там. Данные включают номер врача, номер пациента, порядковый номер приема, название курса лечения, дату назначения, время и примечания.

При необходимости врач запрашивает в медицинском секретариате историю болезни пациента, содержащую данные о курсах лечения, полученных пациентом.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ**)**

Методические рекомендации к выполнению лабораторных работ по курсу «Современные проблемы информатики и вычислительной техники»

Составитель: к.т.н., доцент кафедры ИСТ
Куликовских И.М.

Самара 2013 г.

1. ПОСТРОЕНИЕ ОРТОГОНАЛЬНЫХ МОДЕЛЕЙ КОРРЕЛЯЦИОННО-СПЕКТРАЛЬНЫХ ХАРАКТЕРИСТИК С ПОМОЩЬЮ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ (АИС)

Цель работы: изучение методов и приобретение практических навыков построения ортогональных моделей корреляционно-спектральных характеристик временных рядов.

1.1. Задание на самостоятельную работу

1. Сгенерировать временной ряд с заданным видом корреляционной функции и со следующими параметрами - $M = ent[\tau_{k\,max} / \Delta\tau]$, $N=5000$, $\delta = 0{,}02$.
2. Построить КФ и её фазовый портрет, сравнить с теоретическими кривыми.
3. Для заданного ортогонального базиса определить параметры модели и погрешность аппроксимации.
4. Определить интервалы корреляции, сравнить с теоретическими интервалами, найти относительную погрешность оценивания интервалов корреляции.
5. Найти корректирующие коэффициенты, обеспечивающие условие нормировки ортогональной модели КФ, и построить модель КФ.
6. Определить интервалы корреляции, сравнить с теоретическими интервалами, найти относительную погрешность оценивания интервалов корреляции.
7. Построить модель спектральной плотности мощности.
8. Определить экстремальную частоту, значение СПМ в точке, соответствующей экстремальной частоте, и эквивалентную ширину спектра мощности. Сравнить полученные результаты с теоретическими характеристиками, найти относительную погрешность оценивания обобщенных характеристик.
9. Передать действительную компоненту СПМ в подсистему аппроксимации составляющих СПМ и построить ортогональную модель действительной части СПМ.
10. Построить КФ и сравнить с теоретической.
11. При исследовании других ортогональных базисов необходимо повторить пункты 3 – 10.
12. Построить КФ «идеального» полосового шума.
13. Построить ортогональную модель КФ.
14. Построить ортогональную модель спектральной плотности мощности.
15. Пункты 13 – 14 повторить и выбрать наилучший базис.

1.2. Содержание отчёта

1. Цель работы.
2. Задание.
3. Результаты выполнения работы в автоматизированной системе, представленные в виде экранных форм.
4. Выводы.

1.3. Контрольные вопросы

1. Какой метод положен в основу генерации временных рядов с заданным видом корреляционной функции?
2. Из каких соображений выбирают численное значение интервала дискретизации временного ряда?
3. Какой метод фильтрации применен в лабораторной работе?
4. Чем отличается метод имитационного моделирования от обработки экспериментальных данных?
5. С какой целью строятся фазовые портреты моделей?
6. Какая часть фазового портрета наиболее информативна?
7. Как по виду фазового портрета определить характер корреляционной функции: монотонная, колебательная?
8. Что из себя представляет «идеальный полосовой шум»?

2. ПОСТРОЕНИЕ ОРТОГОНАЛЬНЫХ МОДЕЛЕЙ ВЗАИМНЫХ КОРРЕЛЯЦИОННО-СПЕКТРАЛЬНЫХ ХАРАКТЕРИСТИК С ПОМОЩЬЮ АИС

Цель работы: изучение методов и приобретение практических навыков построения ортогональных моделей взаимных корреляционно-спектральных характеристик временных рядов.

2.1. Задание на самостоятельную работу

16. Сгенерировать СП с заданным видом ВКФ.
17. Построить ВКФ и её фазовый портрет, сравнить с теоретическими кривыми.
18. Для заданного ортогонального базиса определить параметры модели ВКФ и погрешность аппроксимации.
19. Определить интервалы корреляции, сравнить с теоретическими интервалами, найти относительную погрешность оценивания интервалов корреляции.
20. Найти корректирующие коэффициенты, обеспечивающие условие нормировки ортогональной модели ВКФ, и построить модель ВКФ.
21. Определить интервалы корреляции, сравнить с теоретическими интервалами, найти относительную погрешность оценивания интервалов корреляции.
22. Построить модель взаимной спектральной плотности мощности.
23. Передать действительную и мнимую компоненты взаимной СПМ в подсистему аппроксимации составляющих СПМ и построить их ортогональные модели.
24. Построить ВКФ и сравнить с теоретической.

2.2. Содержание отчёта

5. Цель работы.
6. Задание.
7. Результаты выполнения работы в автоматизированной системе, представленные в виде экранных форм.
8. Выводы.

2.3. Контрольные вопросы

1. В чём заключается специфика аппроксимации взаимных корреляционных функций по сравнению с аппроксимацией автокорреляционных функций?
2. Какие численные методы применяются при аппроксимации взаимных корреляционных функций?
3. Из каких соображений выбирается начальное приближение при аппроксимации взаимных корреляционных функций параметрическими моделями?
4. Как отличить фазовый портрет колебательной взаимной корреляционной функции от монотонной?
5. Как определяются корректирующие коэффициенты взаимной корреляционной функции, обеспечивающие условие нормировки?
6. Как определяются интервалы корреляции взаимной корреляционной функции?
7. Назовите основное отличие спектральной плотности мощности и взаимной спектральной плотности мощности.

3. АНАЛИЗ МЕТОДИЧЕСКИХ ПОГРЕШНОСТЕЙ ОЦЕНКИ ПАРАМЕТРОВ ОРТОГОНАЛЬНЫХ МОДЕЛЕЙ КОРРЕЛЯЦИОННО-СПЕКТРАЛЬНЫХ ФУНКЦИЙ С ПОМОЩЬЮ АИС

Цель работы: изучение методики и приобретение практических навыков анализа методических погрешностей оценки параметров ортогональных моделей корреляционно-спектральных функций и их характеристик методом имитационного моделирования.

3.1. Теоретические основы вычислительного практикума (см. CD)

3.2. Задание на самостоятельную работу

1. Задать параметры имитационного моделирования в соответствующих подсистемах автоматизированной системы:
   a. Вид корреляционной функции $\rho_x(\tau,\omega_0/\lambda)$ с параметрами - $M = ent[\tau_{k\,max}/\Delta\tau],\ \mu = \omega_0/\lambda,\ \delta = 0{,}05; 0{,}02; 0{,}1; 0{,}2$.
   b. Вид ортогональной функции $\psi_k(\tau,\gamma)$.
   c. Настройка параметров аппроксимации (выполняемость основного свойства, инвертирование, установка максимума в «нуль», диапазон поиска числа членов разложения).
   d. Объем выборки N.
   e. Число проводимых экспериментов n_{exp}.
2. Провести имитационное моделирование и выгрузить результаты моделирования в MS Excel.

3. Построить следующие зависимости с помощью MS Excel:

a. Графические зависимости $M[\delta](N/\mu, n_{exp})$, $\sigma[\delta](N/\mu, n_{exp})$, $max\{|\delta_j|\}(N/\mu, n_{exp})$, $\delta_{\tau_k^{(2)}}(N/\mu, n_{exp})$, $\delta_{\tau_k^{(4)}}(N/\mu, n_{exp})$, $\delta_{\omega_э^{(1)}}(N/\mu, n_{exp})$, $\delta_{\omega_э^{(2)}}(N/\mu, n_{exp})$, $\delta_{S_x^{(1)}(\omega_э)_{max}}(N/\mu, n_{exp})$, $\delta_{S_x^{(2)}(\omega_э)_{max}}(N/\mu, n_{exp})$, $\delta_{\Delta\omega_э^{(1)}}(N/\mu, n_{exp})$, $\delta_{\Delta\omega_э^{(2)}}(N/\mu, n_{exp})$.

b. Графические зависимости $M[\delta](\mu/N, n_{exp})$, $\sigma[\delta](\mu/N, n_{exp})$, $max\{|\delta_j|\}(\mu/N, n_{exp})$, $\delta_{\tau_k^{(2)}}(\mu/N, n_{exp})$, $\delta_{\tau_k^{(4)}}(\mu/N, n_{exp})$, $\delta_{\omega_э^{(1)}}(\mu/N, n_{exp})$, $\delta_{\omega_э^{(2)}}(\mu/N, n_{exp})$, $\delta_{S_x^{(1)}(\omega_э)_{max}}(\mu/N, n_{exp})$, $\delta_{S_x^{(2)}(\omega_э)_{max}}(\mu/N, n_{exp})$, $\delta_{\Delta\omega_э^{(1)}}(\mu/N, n_{exp})$, $\delta_{\Delta\omega_э^{(2)}}(\mu/N, n_{exp})$.

c. Графические зависимости $M[\delta](n_{exp}/N, \mu)$, $\sigma[\delta](n_{exp}/N, \mu)$, $max\{|\delta_j|\}(n_{exp}/N, \mu)$, $\delta_{\tau_k^{(2)}}(n_{exp}/N, \mu)$, $\delta_{\tau_k^{(4)}}(n_{exp}/N, \mu)$, $\delta_{\omega_э^{(1)}}(n_{exp}/N, \mu)$, $\delta_{\omega_э^{(2)}}(n_{exp}/N, \mu)$, $\delta_{S_x^{(1)}(\omega_э)_{max}}(n_{exp}/N, \mu)$, $\delta_{S_x^{(2)}(\omega_э)_{max}}(n_{exp}/N, \mu)$, $\delta_{\Delta\omega_э^{(1)}}(n_{exp}/N, \mu)$, $\delta_{\Delta\omega_э^{(2)}}(n_{exp}/N, \mu)$.

4. Изменить пункты 1.4 и 1.5.
5. Изменить пункт 1.1.
6. Изменить пункт 1.2, повторить пункты 1 - 5 и выбрать наилучший базис.

3.3. Содержание отчёта

9. Цель работы.
10. Задание.
11. Результаты проведения имитационного моделирования в автоматизированной системе, представленные в виде экранных форм.
12. Результаты проведения имитационного моделирования в виде документа MS Excel.
13. Результаты построения требуемых графических зависимостей в виде документа MS Excel.
14. Выводы.

3.4. Контрольные вопросы

1. Каким образом влияет изменение объема выборки на погрешность аппроксимации? Погрешности оценки обобщенных корреляционно-спектральных характеристик?
2. Каким образом влияет изменение показателя колебательности на погрешность аппроксимации? Погрешности оценки обобщенных корреляционно-спектральных характеристик?

3. Влияет ли на результаты проведенных экспериментов пересчет коэффициентов разложения с учетом выполнения основного свойства?

4. Какое число экспериментов рекомендуется проводить? Почему?

5. Какой из способов оценки интервалов корреляции дает худший результат? Почему?

6. Какой из способов оценки эквивалентной ширины спектральной плотности мощности дает лучший результат? В каких случаях и почему?

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)

ТЕСТЫ ДЛЯ ИТОГОВОГО КОНТРОЛЯ ЗНАНИЙ
по курсу
«Современные проблемы информатики и вычислительной техники»

Составители: д.т.н., доцент кафедры ИСТ
Иващенко А.В.
к.т.н., доцент кафедры ИСТ
Куликовских И.М.

Самара 2013 г.

ФИО _____ № группы_____

1. В IDEF0 система представляется как совокупность взаимодействующих _____

2. В методологии SADT _____ – искусственный объект, представляющий собой отображение (образ) системы и ее компонентов.

3. Соответствие элементов групп. Стрелки SADT

Вход (input)	Ресурсы, которые выполняют работу
Выход (output)	Правила, стратегии, процедуры или стандарты, которыми руководствуется работа
Управление (control)	Материал или информация, которые производятся работой
Механизм (mechanism)	Материал или информация, которые используются или преобразуются работой для получения результата

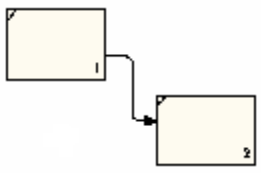
4. Какие графические элементы содержит диаграмма модели SADT?

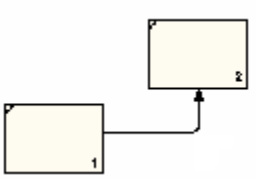
□ Функциональные блоки □ Стрелки
□ Акторы □ Объекты типа «механизм»

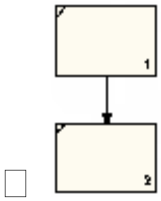
5. Сколько блоков отображается на каждой диаграмме SADT?

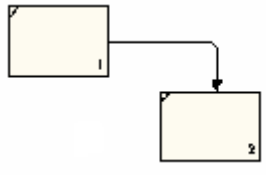
□ 1 – 3 □ 3 – 6 □ 6 – 10 □ 10 – 15

6. Какое(ие) из следующих соединений функциональных блоков неправильно?

□ 

□ 

□ 

□ 

7. Какое преобразование отображает функциональный блок?

□ вход в выход
□ управление в выход
□ вход в управление

8. Какие диаграммы описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией как внутри системы между бизнес-функциями, так и системы в целом с внешней информационной средой?

_____

9. Что является центральными компонентами модели IDEF3

□ Состояния (State) □ Единицы работы Unit of Work (UOW)
□ События (Event) □ Сущности (Entity)

10. Меняется ли идентификатор работы IDEF3

□ присваивается при создании и не меняется никогда
□ задается автором модели
□ присваивается автоматически в случае, если работы с таким идентификатором не существует

11. Укажите наименование перекрестка (IDEF3) : 

□ Асинхронное AND □ Синхронное OR
□ Асинхронное OR □ XOR
□ Синхронное AND

12. Какой перекресток IDEF3 используется в случае, когда:
Все предшествующие (следующие) процессы должны быть завершены (запущены)

□ Асинхронное AND □ Синхронное OR
□ Асинхронное OR □ XOR
□ Синхронное AND

13. Какая из указанных методологий описывает методы разработки реляционных баз данных

□ IDEF1X □ IDEF2X □ IDEF3 □ IDEF4

14. Укажите какие из следующих объектов, используются в рамках нотации ARIS eEPC

□ функция
□ событие
□ компонент
□ организационная единица
□ документ
□ актор
□ кластер информации
□ класс

15. Язык _____ представляет собой унифицированный язык визуального моделирования

16. Отношение обобщения UML отображается как стрелка
□ от предка к потомку □ от потомка к предку
□ не имеет значения

17. Какие из следующих отношений используются на диаграмме вариантов использования? Укажите четыре наиболее часто используемые отношения.
□ ассоциации □ включения
□ агрегации □ расширения
□ композиции □ обобщения

18. Класс на диаграмме классов разделен горизонтальными линиями на несколько блоков, содержащих:
□ наименование □ состояния
□ атрибуты □ методы

19. Какое отношение на диаграмме классов является частным случаем ассоциации (когда один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности)?
□ ассоциации □ включения
□ агрегации □ расширения
□ композиции □ зависимости
□ обобщения

20. На какой диаграмме UML ключевым моментом является динамика взаимодействия объектов во времени (вертикальная временная ось направлена сверху вниз).

_____

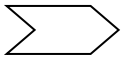
21. Какой вид отношения отображен на данном рисунке?

□ ассоциации □ включения
□ агрегации □ расширения
□ композиции □ зависимости
□ обобщения

22. Какие из следующих представлений входят в ARIS:

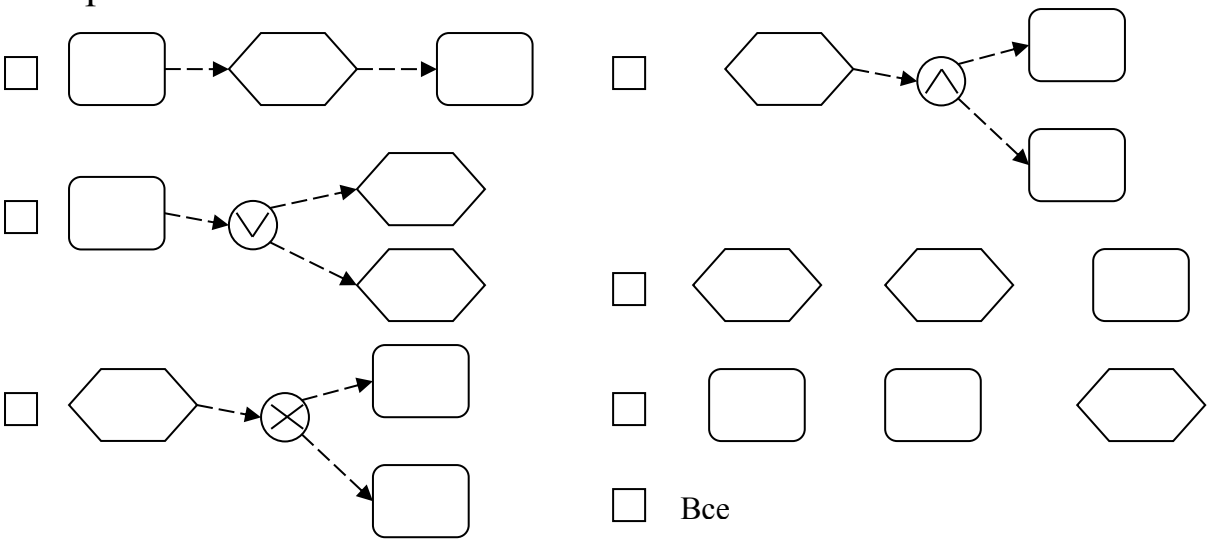
□ операционное □ функциональное □ событийное
□ информационное □ организационное □ управляющее

23. Какая разновидность диаграмм eEPC предусматривает распределение графических элементов по столбцам? _____

24. Какой элемент VAD отображается как 

□ процесс □ ресурс □ событие □ стрелка
□ сущность □ организационная единица □ такого элемента нет

25. Какое(ие) из следующих соединений событий и функций неправильно?



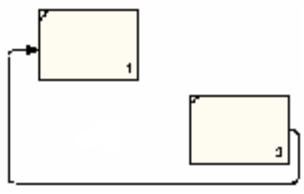
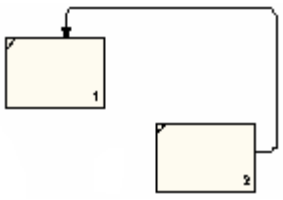
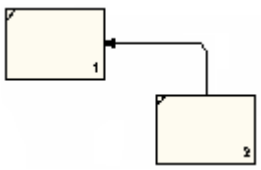
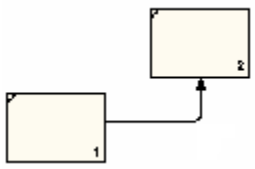
□ Все

ФИО _____ № группы_____

1. Какой стандарт описывает основные положения методологии SADT?
□ IDEF 0 □ IDEF 1 □ IDEF 2 □ IDEF 3
2. С чего начинается построение модели по методологии SADT?
□ Построение контекстной диаграммы □ Определение возможных функций
□ Описание цели моделирования и точки зрения □ Определение правил доминирования
3. Соответствие элементов групп. Расположение стрелок относительно SA блоков

Вход (input)	Верхняя грань
Выход (output)	Правая грань
Управление (control)	Левая грань
Механизм (mechanism)	Нижняя грань

4. _____ обозначает поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты.
5. Когда завершается моделирования
□ Когда декомпозированы блоки на диаграмме декомпозиции верхнего уровня
□ Когда цель проектирования можно считать достигнутой
□ Когда модель соответствует требованиям
6. Какое(ие) из следующих соединений функциональных блоков неправильно?

□ 
□ 
□ 
□ 

7. Какое из следующих утверждений верно для нумерации SA блоков
□ Блок A21 содержится на диаграмме декомпозиции блока A1
□ Блок A21 содержится на диаграмме декомпозиции блока A2
□ Блок A2 содержится на диаграмме декомпозиции блока A21
8. В диаграмме DFD, что указывает на место, организацию или человека, которые участвуют в процессе обмена информацией с системой, но располагаются за рамками данной модели?
_____

9. Что в IDEF3 описывает какой-либо бизнес-процесс – последовательность изменений свойств объекта, в рамках рассматриваемого процесса, например последовательность обработки заказа или события, которые необходимо обработать за конечное время?

□ Работа □ Сценарий □ Последовательность □ Алгоритм

10. Соответствие элементов групп. В IDEF3 различают три типа стрелок, изображающих связи:

Связь предшествования (Precedence) – показывает, что прежде чем начнется работа-приемник, должна завершиться работа-источник.	Сплошная линия
Связь отношения (Relational) – показывает связь между двумя работами или между работой и объектом ссылки.	Стрелка с двумя наконечникам и
Поток объектов (Object Flow) – показывает участие некоторого объекта в двух или более работах	Пунктирная линия

11. Укажите наименование перекрестка (IDEF3) : 

□ Асинхронное AND □ Синхронное AND □ XOR
□ Асинхронное OR □ Синхронное OR

12. Какой перекресток IDEF3 используется в случае, когда:

Все предшествующие (следующие) процессы завершены (запускаются) одновременно

□ Асинхронное AND □ Синхронное OR
□ Асинхронное OR □ XOR
□ Синхронное AND

13. Что в DEF1X описывает конкретный набор экземпляров реального мира?

□ Класс □ Объект □ Таблица
□ Сущность □ Кортеж

14. Какие диаграммы входят в UML?

□ вариантов использования □ последовательности □ компонентов
□ развертывания
□ классов □ цепочек процесса □ программных продуктов
□ рисков □ стоимостного анализа
□ состояний
□ деятельности □ кооперации

15. Как называется любая сущность, взаимодействующая с системой извне на диаграмме вариантов использования

□ Актор □ Внешняя сущность □ Субъект □ Класс

16. Какое отношение на диаграмме вариантов использования служит для обозначения специфической роли актера в отдельном варианте использования?

□ ассоциации □ обобщения □ зависимости
□ агрегации □ включения
□ композиции □ расширения

17. Отношение включения отображается как стрелка

□ от включающего к включаемому варианту использования
□ от включаемого к включающему варианту использования
□ не имеет значения

18. Какие из следующих отношений используются на диаграмме классов? Укажите четыре наиболее часто используемые отношения.

□ ассоциации □ обобщения
□ агрегации □ включения
□ композиции □ расширения

19. Что на диаграмме последовательности отображают вертикальные линии

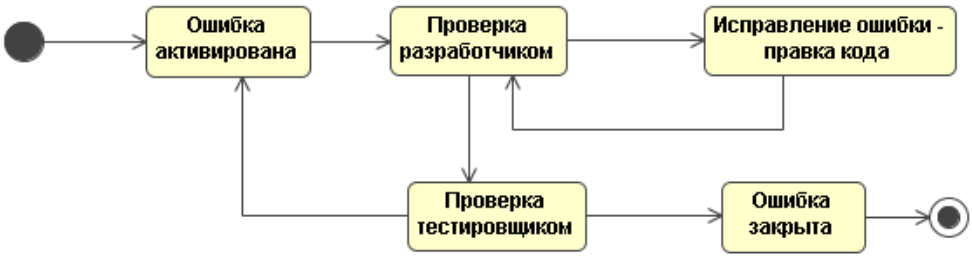
□ активности акторов □ линии жизни классов
□ линии жизни объектов □ сообщения

20. Какое отношение на диаграмме классов является частным случаем агрегации и служит для выделения специальной формы отношения «часть-целое», при которой составляющие части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его части?

□ ассоциации □ обобщения □ зависимости
□ агрегации □ включения
□ композиции □ расширения

21. Как называется данная диаграмма?

□ классов □ последовательности
□ состояний □ кооперации
□ деятельности



22. Сколько представлений входит в ARIS _____

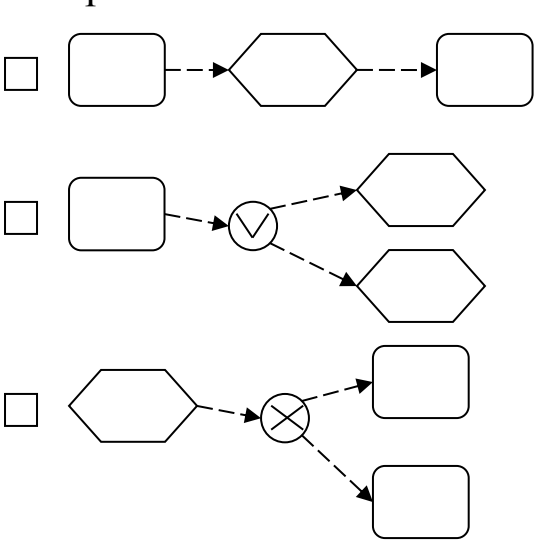
23. В каком порядке располагаются графические элементы на диаграмме eEPC

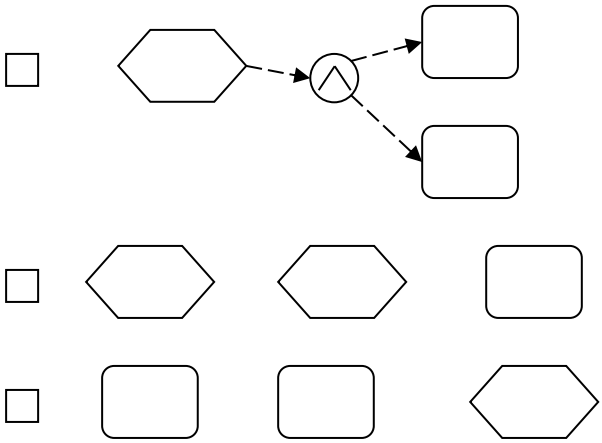
- в порядке доминирования
- в произвольном порядке
- сверху вниз

24. Какой элемент VAD отображается как 

- процесс
- ресурс
- событие
- стрелка
- сущность
- организационная единица
- такого элемента нет

25. Какое(ие) из следующих соединений событий и функций неправильно?





- Все

ФИО _____ № группы_____

1. Что определяет ***расположение*** функциональных блоков на диаграмме SADT?

□ Правила декомпозиции □ Правила доминирования
□ Последовательность выполнения функций □ Сложность функций

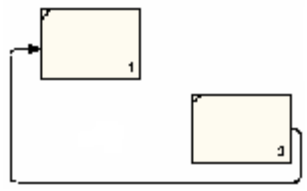
2. _____ _____ является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой.

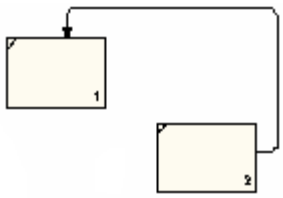
3. Правильное название функционального блока в нотации SADT

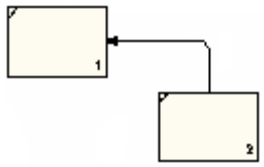
□ глагол или глагольный оборот
□ отглагольное существительное

4. _____ – направленная линия, состоящая из одного или нескольких сегментов, которая моделирует канал, передающий данные от источника к потребителю в методологии SADT

5. Какие варианты обратной связи функциональных блоков возможны в SADT?

□ 

□ 

□ 

□ все указанные варианты

6. Подразумевает ли SADT специальные правила оформления диаграмм

□ Да, имеется рамка IDEF0 □ Нет

7. Как обычно располагаются внешние сущности DFD

□ в центре диаграммы □ по краям диаграммы □ в любом месте

8. Как называется методология моделирования, которая предназначена для описания логики взаимодействия.

_____

9. Что используется в IDEF3 для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы.

□ Операторы
□ События
□ Перекрестки
□ Операторы и перекрестки

10. Укажите наименование перекрестка (IDEF3) : [X]

□ Асинхронное AND
□ Синхронное OR
□ Асинхронное OR
□ XOR
□ Синхронное AND

11. Какой перекресток IDEF3 используется в случае, когда:
Один или несколько предшествующих (следующих) процессов должны быть завершены (запущены)

□ Асинхронное AND
□ Синхронное OR
□ Асинхронное OR
□ XOR
□ Синхронное AND

12. Какое из определений лучше всего подходит для описания нотации ARIS eEPC?

□ расширенная нотация описания цепочки процесса, управляемого событиями
□ расширенная нотация объектно-ориентированного описания бизнес-процессов
□ расширенная нотация описания схемы алгоритма

13. Каким графическим объектом описывается сущность на диаграмме IDEF1X?

□ Прямоугольник □ Овал □ Ромб □ Треугольник

14. Какие диаграммы поведения входят в UML?

□ вариантов использования
□ последовательности
□ компонентов
□ развертывания
□ классов
□ цепочек процесса
□ программных продуктов
□ рисков
□ стоимостного анализа
□ состояний
□ кооперации
□ деятельности

15. На диаграмме вариантов использования актором может быть

□ Человек
□ Техническое устройство
□ Программа или любая другая система
□ Событие
□ Функция

16. Какое отношение на диаграмме вариантов использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями родительских вариантов?

□ ассоциации
□ агрегации
□ композиции
□ обобщения
□ включения
□ расширения
□ зависимости

17. Отношение расширения отображается как стрелка

□ от расширяющего к расширяемому варианту использования
□ от расширяемого к расширяющему варианту использования
□ не имеет значения

18. Какое отношение на диаграмме классов указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, выраженное в том, что некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели

□ ассоциации
□ агрегации
□ композиции
□ обобщения
□ включения
□ расширения
□ зависимости

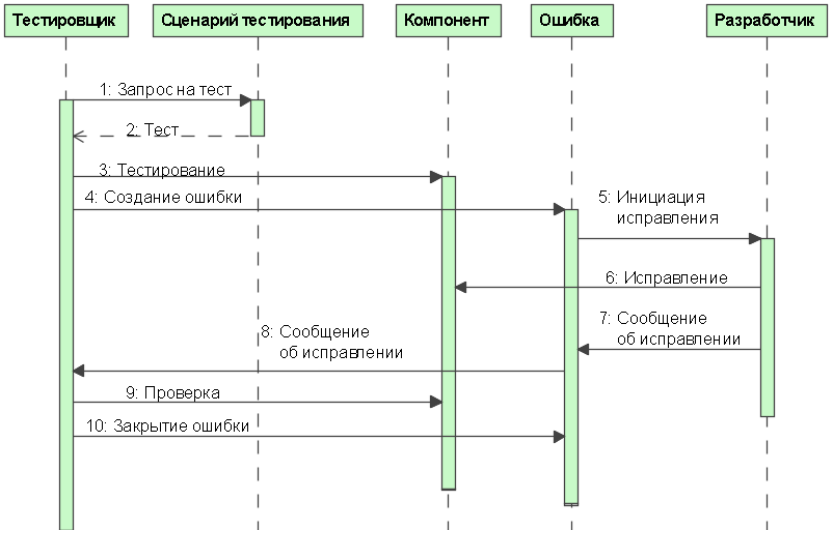
19. Как называется абстрактный метакласс, используемый для моделирования отдельной ситуации; при этом имеет место выполнение некоторого условия.

□ предок □ событие □ состояние □ сущность

20. Какие разновидности сообщений могут встречаться на диаграмме последовательности UML?

□ вызов процедур, выполнение операций или обозначение отдельных вложенных потоков управления
□ обозначение простого (не вложенного) потока управления
□ асинхронное сообщение между двумя объектами в некоторой процедурной последовательности
□ возврат из вызова процедуры.
□ все указанные варианты

21. Как ется данная диаграмма?



- ☐ классов
- ☐ состояний
- ☐ деятельности
- ☐ последовательности
- ☐ кооперации

22. Какие этапы проработки представлений предусматривает ARIS

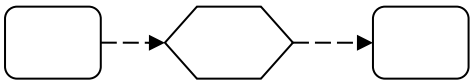
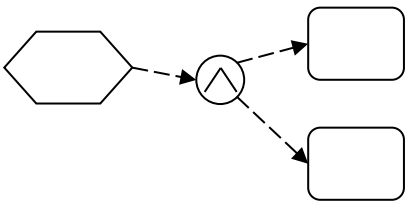
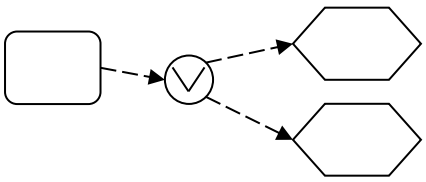
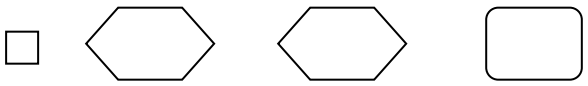
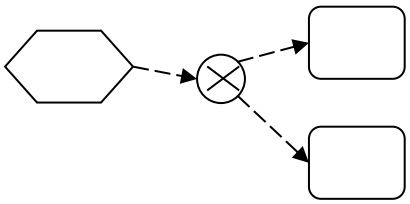
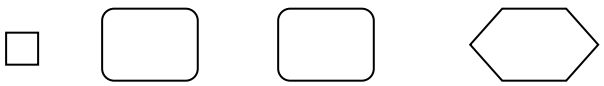
- ☐ описание требований
- ☐ проектирования
- ☐ апробации
- ☐ спецификации
- ☐ реализации
- ☐ документирования

23. В ARIS eEPC _____ служит для отображения возможных результатов выполнения функций и инициируют выполнение новых функций.

24. Что в ARIS отображается как 

- ☐ процесс
- ☐ ресурс
- ☐ событие
- ☐ стрелка
- ☐ сущность
- ☐ организационная единица
- ☐ такого элемента нет

25. Какое(ие) из следующих соединений событий и функций неправильно?

- ☐ 
- ☐ 
- ☐ 
- ☐ 
- ☐ 
- ☐ 
- ☐ Все

ФИО _____ № группы_____

1. Какой стандарт используется для описания модели сущность-связь?

□ IDEF 0 □ IDEF 1x □ IDEF 2 □ IDEF 3

2. Как располагаются функциональные блоки в порядке доминирования?

□ Сверху вниз

□ Слева направо

□ Наиболее доминирующая работа располагается в левом верхнем углу диаграммы, наименее – в правом нижнем

□ Не имеет значения

3. Как называются коды идентификации граничных стрелок?

_____

4. Когда завершается декомпозиция функциональных блоков?

□ Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, удовлетворяющая всех участников проектирования.

□ Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, соответствующая требованиям к модели.

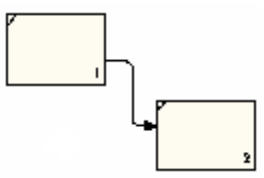
□ Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования.

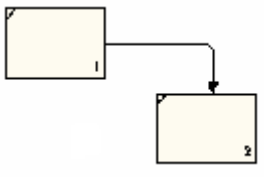
5. Какие из следующих вариантов названия функционального блока в нотации SADT правильные?

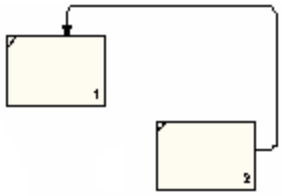
□ Обработать данные □ Формировать отчет

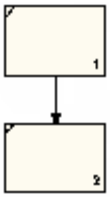
□ Обработка данных □ Отчет

6. Какие варианты передачи управления возможны в SADT?

□ 

□ 

□ 

□ 

7. Какие модели позволяет строить инструментальное средство AllFusion Process Modeler

□ IDEF0 □ IDEF3 □ DFD □ UML

8. Может ли одна внешняя сущность DFD быть использована многократно на одной или нескольких диаграммах?

□ Да □ Нет

9. Правильное название работы IDEF3
□ глагол или глагольный оборот
□ одиночное отглагольное существительное
□ отглагольное существительное одиночное, или в составе фразы

10. Укажите наименование перекрестка (IDEF3) 
□ Асинхронное AND
□ Асинхронное OR
□ Синхронное AND
□ Синхронное OR
□ XOR

11. Какой один из указанных пунктов описывает правила избежания конфликтов в IDEF3:
□ Каждому перекрестку для слияния должен предшествовать перекресток для разветвления;
□ Перекресток для слияния «И» не может следовать за перекрестком для разветвления типа синхронного или асинхронного «ИЛИ», либо исключающего «ИЛИ»;
□ Перекресток для слияния типа исключающего «ИЛИ» не может следовать за перекрестком для разветвления типа «И»;
□ Перекресток, имеющий одну стрелку на одной стороне, должен иметь более одной стрелки на другой.
□ Все указанное

12. Какой перекресток IDEF3 используется в случае, когда:
Один или несколько предшествующих (следующих) процессов завершены (запускаются) одновременно
□ Асинхронное AND
□ Асинхронное OR
□ Синхронное AND
□ Синхронное OR
□ XOR

13. Какой тип связи в IDEF1X устанавливается между независимой и зависимой сущностями?
□ Идентифицирующая
□ Не идентифицирующая

14. Диаграмма _____ служит для представления статической структурной модели системы в терминологии классов объектно-ориентированного проектирования.

15. Какие диаграммы взаимодействия входят в UML?
□ вариантов использования
□ классов
□ рисков
□ состояний
□ деятельности
□ последовательности
□ цепочек процесса
□ стоимостного анализа
□ кооперации
□ компонентов
□ развертывания
□ программных продуктов

16. Какое из следующих определений наилучшим образом подходит? Вариант использования (use case) служит для описания

□ вариантов реализации системы с указанием стоимости

□ сервисов, которые система предоставляет актеру или прецедентов использования системы.

□ требований заказчика и функциональности

□ целей использования системы и ограничений по функциональности

17. Какое из следующих семантических правил описания справедливо для нотации ARIS eEPC

□ логические операторы могут быть определены только между событиями

□ каждая функция должна быть инициирована событием и должна завершаться событием

□ каждая функция должна быть связана с обработкой документа

18. Как на диаграмме классов называется отношение между родителем или предком и более частным и специальным элементом (дочерним или потомком)

□ ассоциации
□ агрегации
□ композиции
□ обобщения
□ включения
□ расширения
□ зависимости

19. Укажите специально выделенные на диаграмме состояний UML частные случаи состояния.

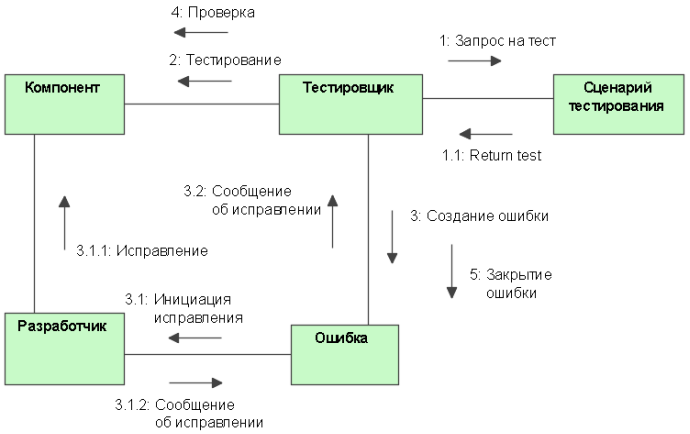
□ начальное □ конечное □ событие □ системное

20. Какая диаграмма, кроме диаграммы последовательности, также отображает взаимодействие

□ Спецификации □ Кооперации □ Коммуникации □ Реализации

21. Как называется данная диаграмма?

□ классов
□ состояний
□ деятельности
□ последовательности
□ кооперации



22. Какие диаграммы ARIS описывают цепочки процессов, добавляющих стоимость, и используются для представления процесса на верхнем уровне. _____

23. Какое из следующих правил справедливо для ARIS eEPC

□ каждая функция должна быть инициирована событием

□ каждая функция должна быть завершена событием

□ каждая функция должна быть инициирована событием и завершена событием

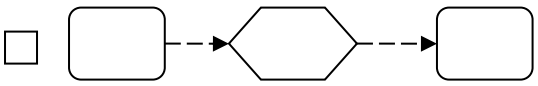
□ последовательность событий и функций произвольно

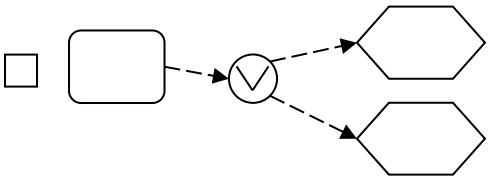
24. Что в ARIS отображается как 

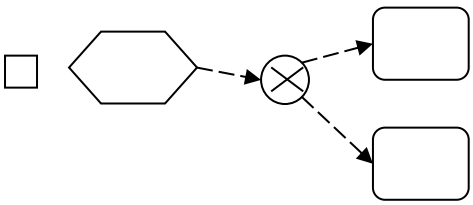
□ процесс □ ресурс □ событие □ стрелка

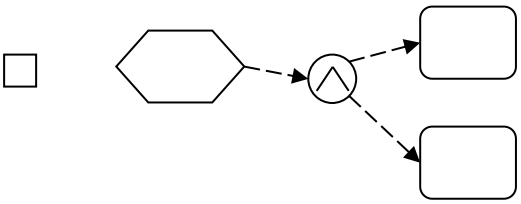
□ сущность □ организационная единица □ такого элемента нет

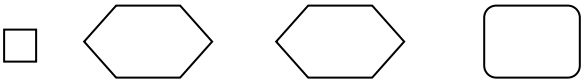
25. Какое(ие) из следующих соединений событий и функций неправильно?

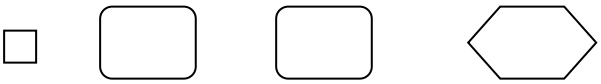
□ 

□ 

□ 

□ 

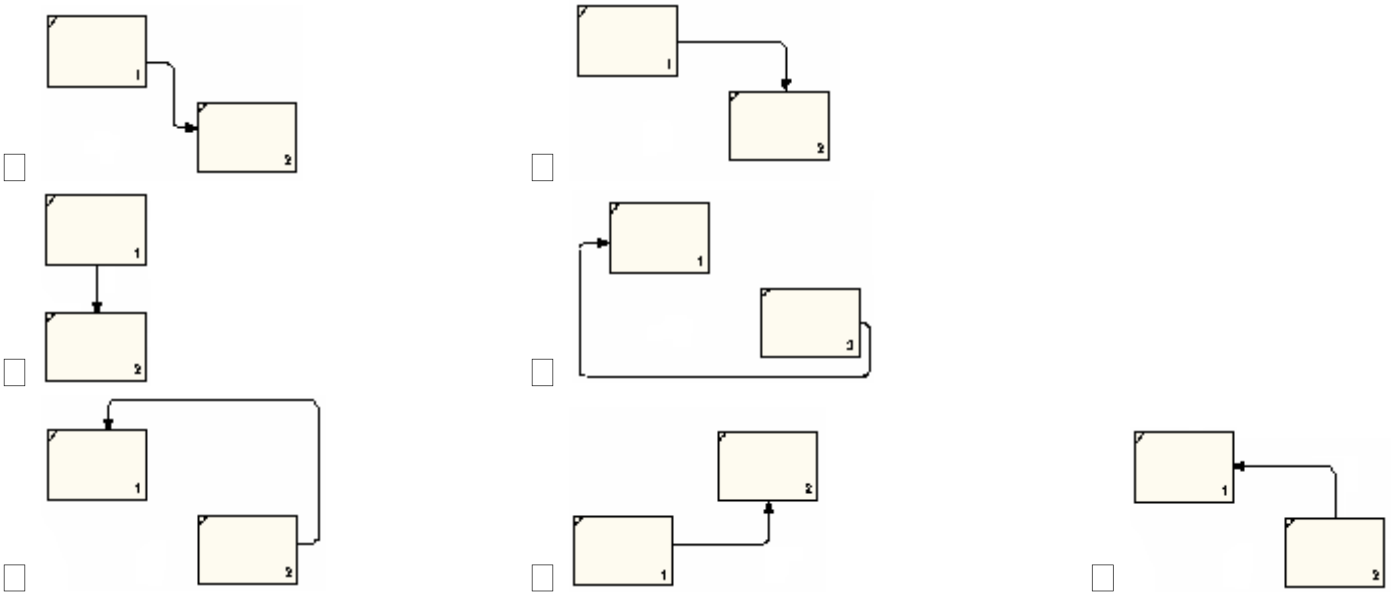
□ 

□ 

□ Все

ФИО _____ № группы_____

1. Как называются диаграммы в методологии Гейна-Сарсона?
□ SADT □ DFD □ IDEF □ CASE
2. _____ в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм, являющихся единицей описания системы
3. Как называются диаграммы, описывают каждый фрагмент и взаимодействие фрагментов модели SADT?
□ взаимодействия □ фрагментов □ детализации □ декомпозиции
4. Какие из следующих вариантов названия стрелки в нотации SADT правильные?
□ Информация об объекте □ Формирование отчета
□ Обработка данных □ Отчет
5. Упорядочить следующие функции преобразующих работ в порядке снижения объема работ:
__ Деятельность __ Процесс __ Операция __ Действие
6. Укажите правильные соединения функциональных блоков стрелками?



7. Возможно ли ветвление стрелок в SADT?
□ Да
□ Для этого нужно использовать специальные операторы
□ Нет

8. Какие объекты DFD позволяют на определенных участках определять данные, которые будут сохраняться в памяти между работами.

□ Внешние сущности
□ Потоки данных
□ Хранилища данных
□ Таблицы

9. Какие из следующих вариантов названия работы IDEF3 правильные?

□ Обработать данные
□ Обработка данных
□ Формировать отчет
□ Отчет

10. Укажите наименование перекрестка (IDEF3) : 

□ Асинхронное AND
□ Асинхронное OR
□ Синхронное AND
□ Синхронное OR
□ XOR

11. Могут перекрестки использоваться как для слияния, так и для разветвления? □ Да □ Нет

12. Какой перекресток IDEF3 используется в случае, когда:
Только один предшествующий (следующий) процесс завершен (запускается)

□ Асинхронное AND
□ Асинхронное OR
□ Синхронное AND
□ Синхронное OR
□ XOR

13. Какой тип связи в IDEF1X устанавливается между двумя независимыми сущностями?

□ Идентифицирующая □ Не идентифицирующая

14. Какие диаграммы реализации входят в UML

□ вариантов использования
□ классов
□ рисков
□ состояний
□ деятельности
□ последовательности
□ цепочек процесса
□ стоимостного анализа
□ кооперации
□ компонентов
□ развертывания
□ программных продуктов

15. Как отображается вариант использования на диаграмме вариантов использования?

□ в виде прямоугольника
□ в виде прямоугольника со скругленными углами
□ в виде овала
□ в виде шестиугольника

16. Какое отношение на диаграмме вариантов использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования?

□ ассоциации □ обобщения □ зависимости
□ агрегации □ включения
□ композиции □ расширения

17. Класс на диаграмме классов отображается в виде

□ в виде прямоугольника □ в виде овала
□ в виде прямоугольника со скругленными углами □ в виде шестиугольника

18. Какое отношение на диаграмме классов соответствует наличию некоторого отношения между классами?

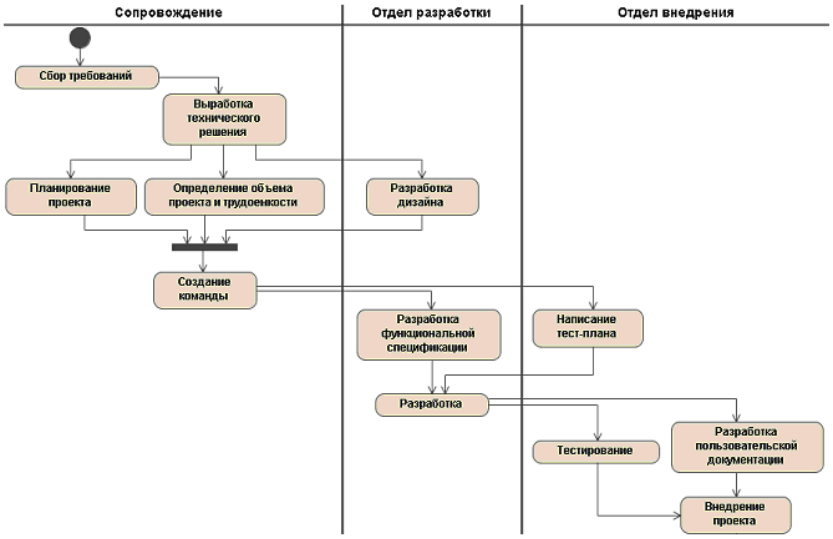
□ ассоциации □ обобщения □ зависимости
□ агрегации □ включения
□ композиции □ расширения

19. Какая диаграмма позволяет описать особенности процедурного и синхронного управления, обусловленного завершением внутренних деятельностей и действий. _____

20. Какие диаграммы описывают реализацию системы

□ Конфигурации
□ Компонентов
□ Развертывания
□ Физического представления

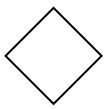
21. Как называется данная диаграмма
- ☐ классов
- ☐ состояний
- ☐ деятельности
- ☐ последовательности
- ☐ кооперации



22. Какие диаграммы ARIS предназначены для описания процессов, управляемых событиями, и содержат события, функции, информационные объекты и организационные единицы _____

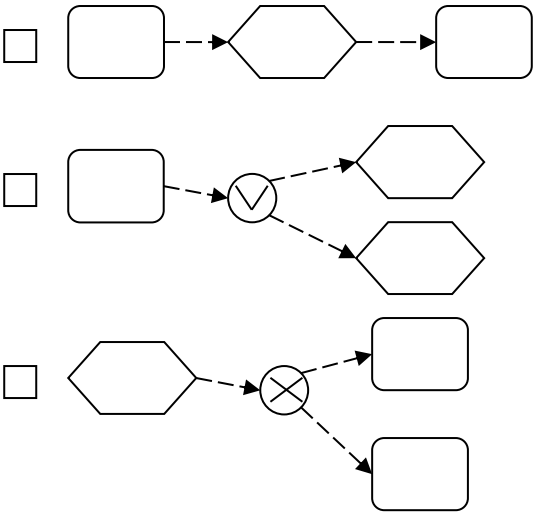
23. Что используется в ARIS eEPC для описания ветвления?

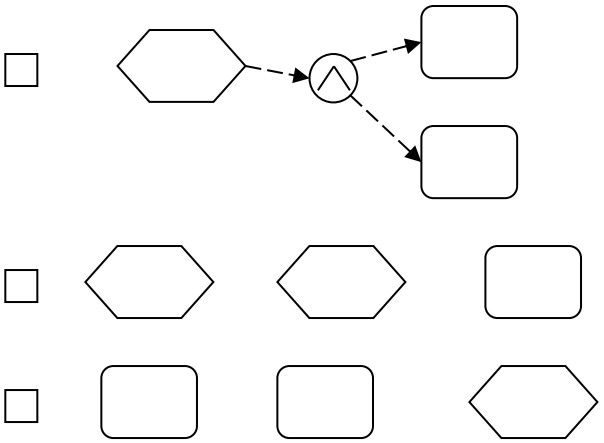
☐ операторы ☐ перекрестки ☐ условия ☐ ветви

24. Что в ARIS отображается как 

☐ процесс ☐ ресурс ☐ событие ☐ стрелка
☐ сущность ☐ организационная единица ☐ такого элемента нет

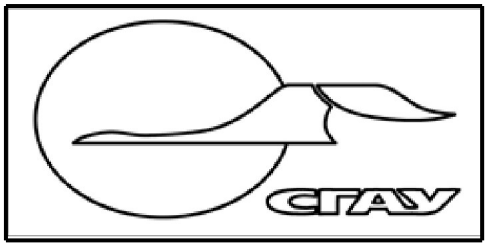
25. Какое(ие) из следующих соединений событий и функций неправильно?





Все

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)



СОГЛАСОВАНО

Управление образовательных программ

_____/ А.В. Дорошин /

"_____"_____ 20___ г.

УТВЕРЖДАЮ

Проректор по учебной работе

_____/ В.Н. Матвеев /

"____" _____ 20___ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование модуля (дисциплины)

Современные проблемы информатики и вычислительной техники

Цикл, в рамках которого происходит освоение модуля (дисциплины)	М2 Профессиональный цикл
Часть цикла	М2.Б.3 Базовая часть

Код учебного плана

230100.68-2013-О-П-2г00м

Факультет	6
Кафедра	"Информационные системы и технологии"

Курс	6		
Семестр	В		
Лекции (СЛ)	18		
Семинарские и практические занятия (СП)	18		
Лабораторные занятия (СЛР)	36	Экзамен	
Контроль самостоятельной работы / Индивидуальные занятия (КСР / ИЗ)	0	Зачет	В
Самостоятельная работа (СРС)	36		
Всего (Всего с экзаменами)	108		

Наименование стандарта, на основании которого составлена рабочая программа:

230100.68 "Информатика и вычислительная техника"

Соответствие содержания рабочей программы, условий ее реализации, материально-технической и учебно-методической обеспеченности учебного процесса по дисциплине всем требованиям государственных стандартов подтверждаем.

Составители:

Куликовских И.М., доцент, к.т.н. _____
(подпись)

Заведующий кафедрой:

Прохоров С.А., профессор, д.т.н. _____
(подпись)

Рабочая программа обсуждена на заседании кафедры

"Информационные системы и технологии"

Протокол №___ от "____" _____ 20___ г.

Наличие основной литературы в фондах научно-технической библиотеки (НТБ) подтверждаем:

Директор НТБ _____ /_____/
(подпись) (расшифровка подписи)

Согласовано:

Декан _____ /_____/
(подпись) (расшифровка подписи)

1 Цели и задачи модуля (дисциплины), требования к уровню освоения содержания

1.1 Перечень развиваемых компетенций

Коды компетенций из ФГОС-3 "Информатика и вычислительная техника" 230100: ОК-6, ПК-1, ПК-2

1.2 Цели и задачи изучения модуля (дисциплины)

Цель и задачи изучения дисциплины:
- понимание проблем создания автоматизированных систем и формирование системного подхода к их решению;
- систематизация знаний о возможностях и особенностях применениянформационных технологий в науке, образовании и технике;
- знание методов, средств, инструментов, применяемых на каждом этапе жизненного цикла программного обеспечения, разрабатываемого в области применения информационных технологий;
- представление об истории развития и формировании современных информационных технологиях и основных парадигм обработки и представлении информации, а также о перспективах развития информационных технологий.

1.3 Требования к уровню подготовки студента, завершившего изучение данного модуля (дисциплины)

Студенты, завершившие изучение данной дисциплины, должны
знать: базовые аспекты информатики и вычислительной техники; современные тенденции в проведении исследований по информатике и вычислительной техники; современные мировые тенденции в разработке новых технических средств автоматизированных систем.
уметь: использовать информационные технологии при решении научных и инженерных задач.

1.4 Связь с предшествующими модулями (дисциплинами)

Для успешного усвоения курса "Современные проблемы информатики и вычислительной техники" студенты должны знать следующие дисциплины:
- интеллектуальные системы;
- технологии проектирования информационно-вычислительных систем;
- моделирование информационно-вычислительных систем.

1.5 Связь с последующими модулями (дисциплинами)

Полученные в результате изучения дисциплины знания могут быть применены в дальнейшем при освоении курсов:
- научно-исследовательская работа магистра.

2 Содержание рабочей программы (модуля)

Семестр 1		

СЛ 0,1667 18 часов 0,5 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 1	Способы представления знаний. Data Mining. Задачи обработки текстовой информации. Классификация. Кластеризация. Метод ближайшего соседа.
		Метод анализа иерархий. Онтологии. Средства построения онтологий. Средства построения онтологий. Системы управления знаниями. Онтологическая СУЗ.
		Семантический Web. Метаданные. Модель метаданных RDF. Язык RDFS.
		Дублинское ядро. Языки онтологий. Язык OWL. Web-2.
		Эволюционные методы. Простой генетический алгоритм. Кроссовер. Примеры применения генетических методов.
		Количество информации. Информационная энтропия. Коэффициент избыточности сообщения. Кодирование информации. Теоремы Шеннона.
		Коды для текстовых документов. Моментальные коды. Сжатие данных. Методы сжатия и форматы данных.
		Методы MPEG. Вейвлеты. Вейвлет-преобразование
		Теории эволюции. Динамические системы. Термодинамическая энтропия. Диссипативные структуры.
		Хаотические системы. Бифуркации. Фракталы. Самоорганизация. Синергетика. Теория катастроф.
		Развитие систем управления предприятиями. Системы управления бизнес-процессами. Архитектурное проектирование систем. Объектно-ориентированное программирование. Компонентно-ориентированные технологии.
		Сетевые службы. Сервис-ориентированная архитектура. Разработка, управляемая моделями. Рефакторинг. Паттерны проектирования. Метамодель.

		Интегрированные среды разработки приложений. Способы интеграции информационных систем. WorkFlow. Технология SOAP. Стандарт UDDI. Язык WSDL. Средства интеграции MCAD и ERP. BizTalk Server.
СП 0,1667 18 часов 0,5 ЗЕТ	Активные 1	Методология SADT. Стандарт IDEF0 Диаграммы потоков данных DFD Моделирование бизнес-процессов. Стандарт IDEF3 Построение ER модели данных. Стандарт IDEF1X.
		Объектно-ориентированное проектирование на языке UML Диаграмма вариантов использования Диаграмма классов Диаграмма состояний Диаграмма деятельности.
		Объектно-ориентированное проектирование на языке UML Диаграмма последовательности Диаграмма кооперации Диаграмма компонентов Диаграмма развертывания.
	Интерактивные 0	
	Традиционные 0	
СЛР 0,3333 36 часов 1 ЗЕТ	Активные 0	
	Интерактивные 1	Построение случайных функциональных характеристик с помощью автоматизированной информационной системы. Построение и реализация математических моделей.
		Обработка реальных данных с помощью автоматизированной информационной системы специального назначения. Интерпретация полученных результатов.
	Традиционные 0	
КСР 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 1	
СРС 0,3333 36 часов 1 ЗЕТ	Активные 0	

	Интерактивные 1	Синергетика – новое научное междисциплинарное направление.
		Облачные вычисления.
		Современные телекоммуникационные системы и технологии.
		Архитектурные особенности и области применения современных процессоров цифровой обработки сигналов.
		Системы интеллектуального анализа данных. Методология Data Mining.
	Традиционные 0	

3 Инновационные методы обучения

1. Проведение практических занятий с применением новейших информационных технологий.
2. Использование в качестве лекционного материала новых научных результатов
2. Обсуждение результатов самостоятельной работы на практических занятиях.

4 Технические средства и материальное обеспечение учебного процесса

1. Класс с проектором и компьютером, работающим под ОС Windows.
2. Компьютерный класс (ОС Windows), наличие прикладного математического программного обеспечения Mathcad 14 и выше, Visual Studio 2008 и выше, IDE NetBeans, Rational Rose (UML), ERWin.

5 Учебно-методическое обеспечение

5.1 Основная литература

1. Проектирование автоматизированных систем обработки информации и управления (АСОИУ) [Текст] : [учеб. по специальности "Автоматизир. системы обраб. информ. и упр." направления подгот. дипломир. специалистов "Информатика и вычисл. техника"] / Я. А. Хетагуров. - М. : Высш. шк., 2006. - 223 с. (21 экз.)

5.2 Дополнительная литература

1. Современные технологии в задачах управления, автоматики и обработки информации [Текст] : тр. XVII Междунар. науч.-техн. семинара, Алушта, сент. 2008 г. / Моск. авиац. ин-т (гос. техн. ун-т) "МАИ" [и др. ; редкол.: Лебедев Г. Н. и др.]. - СПб. : ГУАП, 2008. - 285 с. (1 экз.)

5.3 Электронные источники и интернет ресурсы

1. Прохоров, С.А. Комплекс обучающих программ по курсу «Проектирование АСНИ» [Электронный ресурс] / С. А. Прохоров, И. М. Куликовских; Федер. агентство по образова-нию, Самар. гос. аэрокосм. ун-т им. С. П. Королева, Нац. исслед. ун-т. - Электрон. текстов
ые дан. - Самара : [б. и.], 2009. - 1 эл. опт. диск (CD-RW). - (Программа развития

государственного образовательного учреждения высшего профессионального образования «Самарский государственный аэрокосмический университет имени академика С. П. Королева» на 2009-2018 годы)
2. http://www.computerworld.com/
3. http://www.ssau.ru/resources/sotrudniki/prohorov/11/

5.4 Методические указания и рекомендации

Текущий контроль знаний студентов в каждом семестре завершается на отчетном занятии, результатом которого является допуск или недопуск студента к зачету по дисциплине. Основанием для допуска к зачету является выполнение лабораторных работ и прием индивид
уального задания. Неудовлетворительная оценка не лишает студента права сдавать экзамен и является основанием для дополнительного вопроса (задания) на экзамене.
Промежуточный контроль знаний студентов проводят в виде зачета. Зачет проводится согласно положению о текущем и промежуточном контроле знаний студентов, утвержденному ректором университета.