МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ЧЕБОКСАРСКИЙ ИНСТИТУТ (ФИЛИАЛ) МОСКОВСКОГО ПОЛИТЕХНИЧЕСКОГО УНИВЕРСИТЕТА

Кафедра информационных технологий и систем управления

Методические указания по написанию, оформлению и защите курсовых проектов по дисциплине «Проектирование автоматизированных систем»

Направление подготовки	**27.03.04 «Управление в технических системах»**
	(код и наименование направления подготовки)
Направленность (профиль) подготовки	**«Управление и информатика в технических системах»**
	(наименование профиля подготовки)
Квалификация выпускника	**бакалавр**
Форма обучения	**очная, заочная**

Чебоксары, 2024

Методические рекомендации для обучающихся по подготовке и оформлению курсовых проектов по дисциплине Проектирование автоматизированных систем, по направлению подготовки 27.03.04 Управление в технических системах.

Учебно-методическое пособие. – Чебоксары: Чебоксарский институт (филиал) Московского политехнического института, 2024. – 32 с.

Одобрено кафедрой «Информационных технологий и систем управления». протокол № 8 от 16.03.2024 г.

В Методических рекомендациях изложены методология и методика подготовки курсовых проектов по управлению в технических системах, а также требования к их оформлению; кроме того, определены основные обязанности кафедры информационных технологий и систем управления и научных руководителей по руководству, даны рекомендации студентам по их защите.

Методические рекомендации предназначены для руководителей курсовых проектов, а также для студентов всех форм обучения обучающихся по направлению по направления подготовки 27.03.04 «Управление в технических системах» в Чебоксарском институте (филиале) Московского политехнического университета

Оглавление

Порядок оформления и защиты курсового проекта

Курсовой проект – заключительный этап изучения дисциплины. Цель проекта – систематизация и закрепление теоретических знаний, полученных за время обучения, а также приобретение и закрепление навыков самостоятельной работы. Проект, как правило, основывается на обобщении выполненных студентом лабораторных работ или представляет собой индивидуальное задание по изучаемой дисциплине и подготавливается к защите в завершающий период теоретического обучения.

Тематика курсового проекта определяется преподавателем кафедры. При этом выбор основывается на государственном образовательном стандарте специальности 27.03.04 «Управление и информатика в технических системах». Студенту предоставляется право выбора одной из предложенных тем или предложения своей темы с обоснованием целесообразности ее разработки.

Курсовой проект должен быть подготовлен к защите в срок, устанавливаемый кафедрой и деканатом. К защите курсового проекта представляется:

– пояснительная записка;
– электронная реализация в виде программы и базы данных.

Пояснительная записка содержит основной текст (собственно работа), графические материалы (иллюстрации) и, при необходимости, приложения – разработанную программу с исходным текстом на бумажном носителе, исходные данные и результаты расчетов, алгоритмы, модели, структуры.

Пояснительная записка включает следующие компоненты:

- титульный лист;
- оглавление, включающее наименование всех разделов и пунктов с указанием номеров страниц;
- техническое задание;
- введение, в котором обосновывается актуальность темы, указываются цели и задачи проектирования;
- теоретическую часть, в которой обосновывается выбранный метод решения или модель и полученные закономерности или содержатся описания примененных в работе алгоритмов, структур данных;
- исследовательскую часть, содержащую структуры и исходные данные, полученные результаты (исследования) и их анализ;
- заключение с краткими выводами по результатам работы и предложениями по их использованию;
- список литературы.

Курсовой проект может выполняться как на кафедре, так и в других организациях.

Периодический контроль над работой студента осуществляется преподавателем в процессе проведения консультаций.

Текст работы оформляется в виде пояснительной записки в соответствии с требованиями ГОСТ 2.105.95 «Общие требования к текстовым документам» в объеме 8-20 страниц формата А4. Изложение должно быть последовательным, логичным, конкретным.

Работа оформляется с использованием текстового редактора Word и распечатывается на принтере. Текст пояснительной записки к курсовому проекту делится на разделы, подразделы и пункты. Размещение текста – с одной стороны листа. Размер шрифта – 14, интервал – полуторный, поля слева– 25мм, сверху–15мм, справа – 10 мм, снизу – 20 мм. Нумерация страниц – внизу по середине. Первая страница – титульный лист, далее – оглавление, задание, введение, текст, список литературы (номера первых двух страниц не указываются).

Для вставки формул рекомендуется использовать редактор формул. Формулы нумеруются в пределах каждого раздела, номер указывается справа от формулы – у правой границы текста, в круглых скобках (например, (3.6) – шестая формула в третьем разделе).

Для создания иллюстраций используются графические редакторы или средства графики математических и статистических либо других пакетов. Таблицы могут быть созданы непосредственно в текстовом редакторе или вставлены из прикладной программы. Таблицы и рисунки должны быть пронумерованы и подписаны.

Ссылки на литературные источники указываются в квадратных скобках. При ссылке на информацию, полученную в Internet, указывается соответствующий электронный адрес. Список литературы, использованной при выполнении работы, приводится в конце текста.

Оформленный курсовой проект представляется студентом преподавателю для просмотра в соответствии с учебным планом.

Во время защиты курсового проекта студент должен кратко сформулировать цель работы, изложить содержание, акцентируя внимание на наиболее важных и интересных с его точки зрения решениях, в первую очередь, принятых студентом самостоятельно. При выступлении может быть использована демонстрация созданного программного обеспечения.

Результаты работы оцениваются с учетом качества ее выполнения и ответов на вопросы. При неудовлетворительной оценке работы преподаватель устанавливает, может ли студент представить к повторной защите ту же работу с необходимой доработкой или должен разработать новую тему.

Требования к содержанию работы

В курсовом проекте должны быть отражены следующие этапы жизненного цикла информационной системы (ИС) для исследуемой предметной области (ПО):

– анализ предметной области и построение моделей с использованием нотаций IDEF0, DFD, IDEF3 методологий;

– инфологическое проектирование некоторой подсистемы ИС ПО на базе DFD модели;

– логическое проектирование подсистемы ИС ПО с использованием средств SQL;

– физическое проектирование подсистемы ИС ПО с использованием средств СУБД MS SQL Server.

Требования к IDEF0 модели

Методология функционального моделирования IDEF0 – это технология описания системы в целом как множества *взаимосвязанных* действий или функций. При этом функции системы исследуются независимо от объектов, которые обеспечивают их выполнение.

IDEF0 модель ПО должна включать:

– *Название модели;*

– *Цель модели;*

– *Точку зрения;*

– *Список данных* – указать данные, которые относятся к входам, выходам, управлению, механизмам;

– *Список функций* – иерархический список функций (**не менее 4-х функций первого уровня**) в виде:

Название функции проектируемой системы – A0.

Название функции (первого уровня) – A1.

Название функции (второго уровня) - A11.

Название функции A12.

Название функции A13.

Название функции – A2.

Название функции A21.

Название функции A22.

Название функции A23.

Словарь данных – создается для упрощения понимания создаваемой модели и исключения неоднозначности трактования модели;

Описание функциональных блоков;

Диаграммы модели:

1. Контекстная диаграмма А-0 (диаграмма верхнего уровня) с описанием цели системы и точки зрения модели (пример на рис. 1);
2. Диаграмма декомпозиции А0 (пример на рис. 2);
3. Диаграмма декомпозиции некоторого функционального блока диаграммы А0 (пример на рис 3);
4. Диаграмма дерева узлов (пример на рис. 4).

Диаграммы должны быть выполнены на специальном бланке либо вручную, либо с использованием BPWin (рис.2, 3). Функции и данные, отображаемые на диаграммах, должны быть описаны (рекомендуется выполнить также при помощи BPWin).

Краткая справка по синтаксису IDEF0 диаграмм*

Каждая диаграмма содержит блоки и дуги (стрелки). Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними (рис.2). Диаграмме дается название, которое располагается в центре нижней части ее бланка. На каждой диаграмме написана стандартно идентифицирующая ее информация: автор диаграммы, частью какого проекта является работа, дата создания или последнего пересмотра диаграммы, статус диаграммы. Вся идентифицирующая информация располагается в верхней части бланка диаграммы.

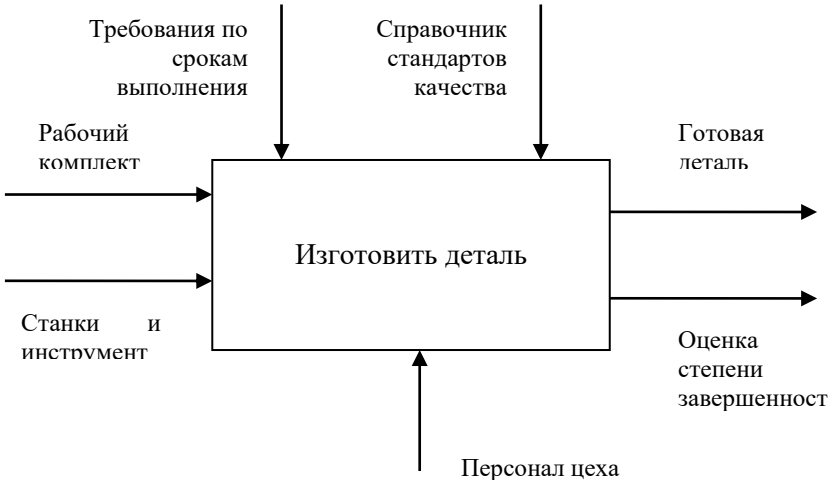

Рисунок1. Контекстная диаграмма А-0 системы «Экспериментальный цех механообработки». Функция (цель) системы – Изготовить деталь.

* см. также http://vmk.ugatu.ac.ru/book/ross/index.html

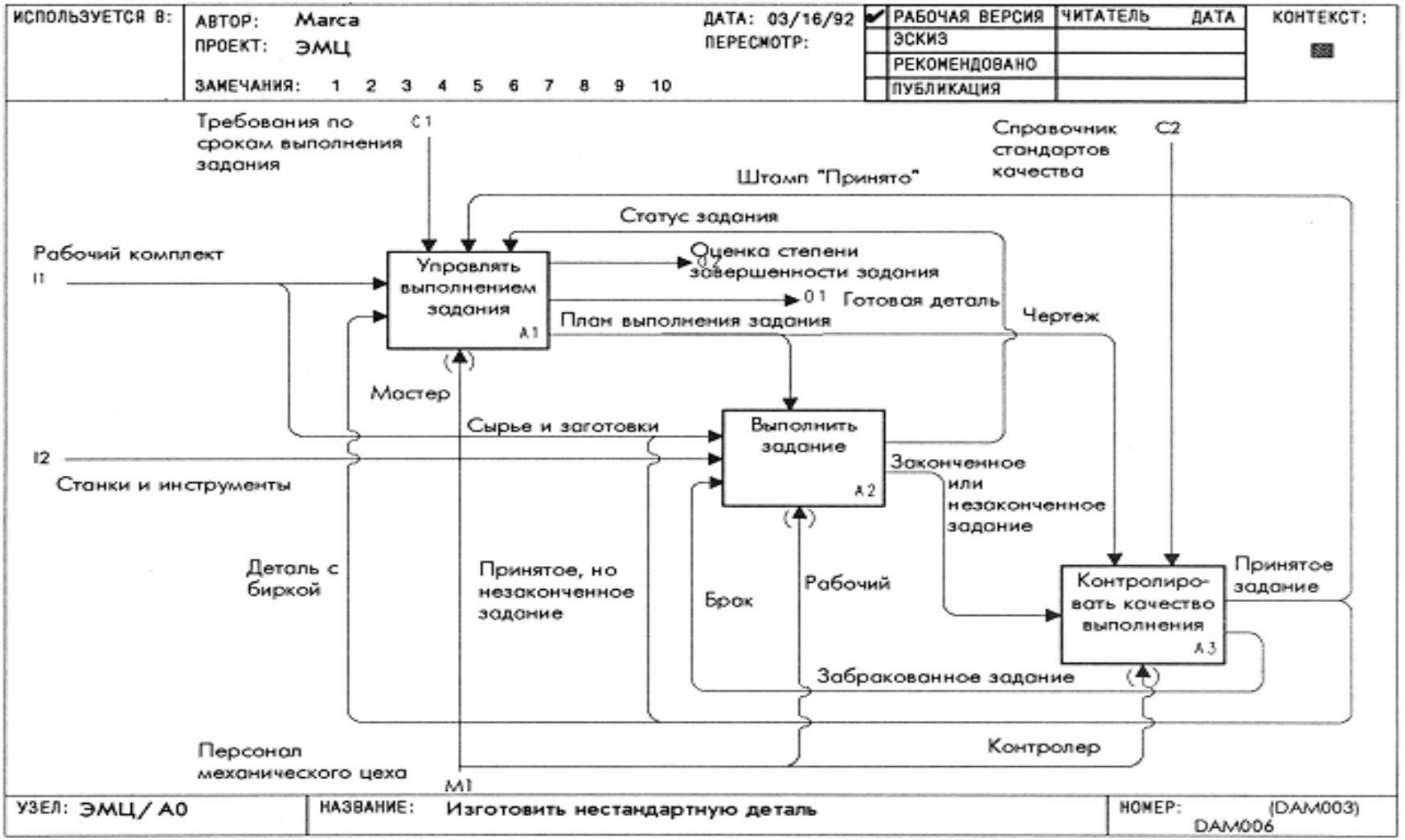

Рисунок 2. Диаграмма декомпозиции A0

Функциональные блоки на диаграммах изображаются прямоугольниками. Блок представляет функцию или активную часть системы, поэтому названиями блоков служат глаголы или глагольные обороты. Например, названиями блоков диаграммы «Выполнить задание» (рис. 2) являются: *определить степень выполнения задания, выбрать инструменты, подготовить рабочее место, обработать на станке и собрать* (рис. 3).

В отличие от других графических методов структурного анализа каждая сторона блока имеет особое, вполне определенное назначение. Левая сторона блока предназначена для входов, верхняя - для управления, правая - для выходов, нижняя - для механизмов. Такое обозначение отражает определенные системные принципы: входы преобразуются в выходы, управление ограничивает или предписывает условия выполнения преобразований, механизмы показывают, кто, что и как выполняет функция. Например, четвертый блок диаграммы «Выполнить задание» может быть интерпретирован следующим образом: *детали, сырье и брак, обрабатываются на станке* и *собираются* в *результаты обработки* с использованием *оборудованного рабочего места.*

Блоки никогда не размещаются на диаграмме случайным образом. Они размещаются по степени важности, как ее понимает автор диаграммы. Этот относительный порядок называется доминированием. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. Например, самым доминирующим блоком диаграммы может быть либо первый из требуемой последовательности функций, либо планирующая или контролирующая функция, влияющая на все другие функции (такая, как *определить степень выполнения задания* на рис. 3).

Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы, а наименее доминирующий - в правом нижнем углу. В результате получается «ступенчатая» схема, подобная представленной на рис. 3 для блоков 1,2,3. Расположение блоков на странице отражает авторское определение доминирования. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные.

Блоки в модели должны быть пронумерованы. Номера блоков служат однозначными идентификаторами для системных функций и автоматически организуют эти функции в иерархию модели. Используя номера блоков и оценивая влияние, которое один блок оказывает на другой, аналитик может организовать модель по принципу функционального доминирования. Это позволяет согласовать иерархический порядок функций в модели с уровнем влияния каждой функции на остальную часть системы.

Дуги на диаграмме изображаются одинарными линиями со стрелками на концах. Для функциональных диаграмм дуга представляет множество объектов. Понятие «объекты» достаточно общее, поскольку дуги могут представлять, например, планы, данные в компьютерах, машины и информацию. Дуги диаграммы «Выполнить задание» на рис. 3 представляют материалы, написанные на бумаге (например, *следующий шаг задания),* физические

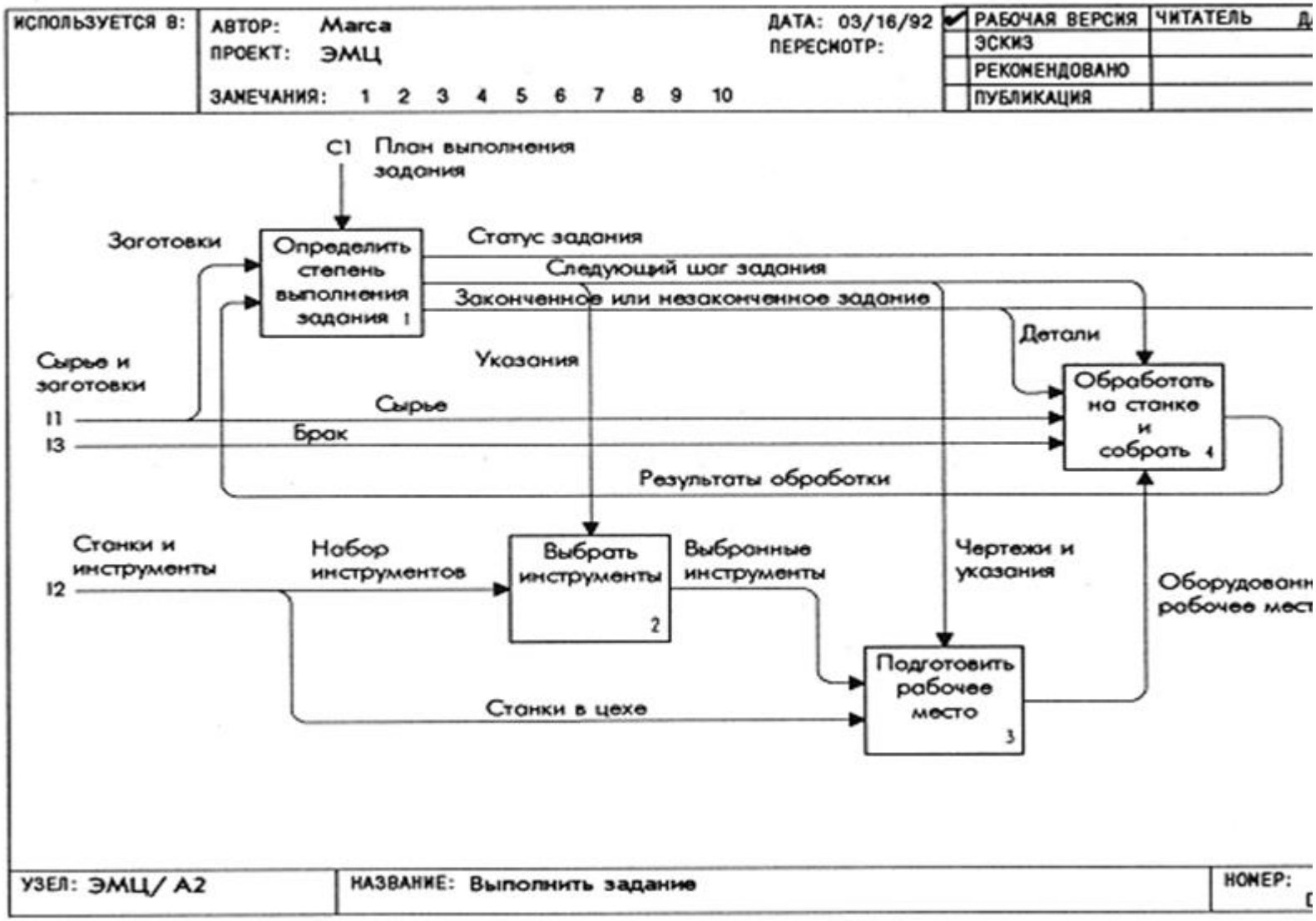

Рисунок 3. Диаграмма декомпозиции A2 «Выполнить задание»

материалы (например, *сырье и заготовки),* инструменты (например, *набор инструментов),* рабочие чертежи (например, *чертежи и указания),* рабочую среду (например, *оборудованное рабочее место)* и управленческую информацию (например, *статус задания).* Однако в системном анализе вместо термина «объекты» часто употребляют термин «данные».

Так как дуги изображают объекты, они описываются (помечаются) существительными или существительными с определениями, располагающимися достаточно близко к линии дуги. Рекомендуется размещать описания дуг, называемые метками, как можно ближе к линиям дуг, не нарушая, однако, читабельность диаграмм. Это устраняет неопределенность в том, к какой дуге относится метка, и исключается необходимость в дополнительных графических связях.

Между объектами и функциями возможны четыре отношения: вход, управление, выход, механизм. Каждое из этих отношений изображается дугой, связанной с определенной стороной блока. Дуги кодируются: код состоит из латинской буквы I, C, O, M и числа (рис. 2, 3). По соглашению левая сторона блока предназначена для входных дуг, верхняя сторона - для управленческих дуг, правая сторона - для выходных дуг, нижняя сторона - для дуг механизмов. Таким образом, стороны блока чисто графически сортируют объекты, изображаемые касающимися блока дугами.

Входные дуги изображают объекты, используемые и преобразуемые функциями. Например, в процессе изготовления детали *сырье* трансформируется функцией *обработать на станке и собрать.* Управленческие дуги представляют информацию, управляющую действиями функций. Обычно управляющие дуги несут информацию, которая указывает, что должна выполнять функция. Например, *следующий шаг задания* определяет, какие нужно выбрать инструменты, какие потребуются станки и цеха и как инструменты и станки должны использоваться при изготовлении детали. Выходные дуги изображают объекты, в которые преобразуются входы. Например, *обработать на станке и собрать* преобразует *сырье* и *брак* в *результаты обработки,* которые в конечном итоге становятся деталями. Дуги механизмов отражают, по крайней мере, частично, как функции (т.е. функции системы) реализуются. Например, *подготовить рабочее место* организует *инструменты и станки* в эффективное пространство для следующего шага задания. Это - рабочая среда, называемая *оборудованным рабочим местом.* Она обозначает место, где рабочий изготавливает деталь, реализуя функцию *обработать на станке и собрать.* Таким образом, *механизмы* изображают физические аспекты функции (склады, людей, организации, приборы).

Итак, диаграмма составлена из блоков, связанных дугами, которые определяют, как блоки влияют друг на друга. Это влияние может выражаться либо в передаче выходной информации к другой функции для дальнейшего преобразования, либо в выработке управляющей информации, предписывающей, что именно должна выполнять другая функция. Например, блок *обработать на станке и собрать* влияет на блок *определить степень выполнения задания,* выдавая ему *результаты обработки* для оценки, а блок

определить степень выполнения задания влияет на очередную операцию блока *обработать на станке и собрать* с помощью *следующего шага задания.* Другими словами, существует сильная управляющая связь блока *определить степень выполнения задания* с блоком *обработать на станке и собрать* и наряду с ней более слабая связь по входу-выходу от блока *обработать на станке и собрать* к блоку *определить степень выполнения задания.*

В методологии IDEF0 требуется только пять типов взаимосвязей между блоками для описания их отношений: управление, вход, обратная связь по управлению, обратная связь по входу, выход-механизм. Связи по управлению и входу являются простейшими, поскольку они отражают прямые воздействия, которые интуитивно понятны и очень просты. Отношение управления возникает тогда, когда выход одного блока непосредственно влияет на блок с меньшим доминированием. Например, блок *определить степень выполнения задания* влияет на блок *выбрать инструменты* в соответствии с детальными *указаниями,* содержащимися в описании *следующего шага задания.* Отношение входа возникает тогда, когда выход одного блока становится входом для блока с меньшим доминированием, например, выход блока *определить степень выполнения задания,* называемый *законченное или незаконченное задание,* становится входом функции *обработать на станке и собрать* при выполнении *следующего шага задания.*

Обратная связь по управлению и обратная связь по входу являются более сложными, поскольку они представляют итерацию или рекурсию. А именно выходы из одной функции влияют на будущее выполнение других функций, что впоследствии влияет на исходную функцию. Обратная связь по управлению возникает тогда, когда выход некоторого блока влияет на блок с большим доминированием. Рассмотрим для примера диаграмму «Изготовить нестандартную деталь» на рис. 2. Функция *управлять выполнением задания* ограничивает действие функции *контролировать качество выполнения* с помощью *чертежа,* в котором указаны разрешенные допуски. Кроме того, дуга *штамп «принято»,* являющаяся выходом блока *контролировать качество выполнения,* организует работу блока *управлять выполнением задания,* поскольку именно *штамп «принято»* указывает, что задание завершено. Таким образом, *штамп «принято»* влияет на будущую деятельность блока *управлять выполнением задания,* поэтому соответствующая дуга направлена назад. Связь по входной обратной связи имеет место тогда, когда выход одного блока становится входом другого блока с большим доминированием. Например, задания, отвергнутые функцией *контролировать качество выполнения,* отсылаются на вход блока *выполнить задание* в качестве *брака.*

Связи «выход-механизм» встречаются нечасто и представляют особый интерес. Они отражают ситуацию, при которой выход одной функции становится средством достижения цели для другой. Например, на рис. 3. представлена функция *подготовить рабочее место,* имеющая выход *оборудованное рабочее место,* который, в свою очередь, является механизмом для блока *обработать на станке и собрать.* Это означает, что *оборудованное рабочее место* необходимо для того, чтобы начать процесс обработки. А в этом

случае дуга механизма обозначает строго последовательную взаимосвязь: приготовления должны быть завершены до начала работы. Поэтому связи «выход-механизм» характерны при распределении источников ресурсов (например, требуемые инструменты, обученный персонал, физическое пространство, оборудование, финансирование, материалы).

Дуга редко изображает один объект. Обычно она символизирует набор объектов. Например, дуга, именуемая *рабочий комплект,* отражает *техническое задание, чертеж, план-график,* некоторое *сырье и заготовки.* Так как дуги представляют наборы объектов, они могут иметь множество начальных точек (источников) и конечных точек (назначений). Поэтому дуги могут разветвляться и соединяться различными сложными способами. Вся дуга или ее часть может выходить из одного или нескольких блоков и заканчиваться в одном или нескольких блоках, как, например, дуга *принятое задание* на рис. 3. Отметим, как различные компоненты дуги *принятое задание* следуют в другие блоки диаграммы: *штамп «принято»* является управляющей информацией для блока *управлять выполнением задания,* в то время как *принятое, но незаконченное задание* является входом в блок *выполнить задание*.

Разветвления дуг, изображаемые в виде расходящихся линий, означают, что все содержимое дуг или его часть может появиться в каждом ответвлении дуги. Дуга всегда помечается до разветвления, чтобы дать название всему набору. Кроме того, каждая ветвь дуги может быть помечена или не помечена в соответствии со следующими правилами:

– непомеченные ветви содержат все объекты, указанные в метке дуги перед разветвлением (т.е. все объекты принадлежат этим ветвям);

– ветви, помеченные после точки разветвления, содержат все объекты или их часть, указанные в метке дуги перед разветвлением (т.е. каждая метка ветви уточняет, что именно содержит ветвь).

Например, на диаграмме «Изготовить нестандартную деталь» дуга *принятое задание* включает несколько объектов и разветвляется в нескольких направлениях. Дуга *штамп "принято "* влияет на блок *управлять выполнением, задания; принятое, но незаконченное задание* идет в блок *выполнить задание* для следующей обработки, а *деталь с биркой* идет в блок *управлять выполнением задания* для окончательной проверки и поставки.

Слияние дуг в SADT, изображаемое как сходящиеся вместе линии, указывает, что содержимое каждой ветви идет на формирование метки для дуги, являющейся результатом слияния исходных дуг. После слияния результирующая дуга всегда помечается для указания нового набора объектов, возникшего после объединения. Кроме того, каждая ветвь перед слиянием может помечаться или не помечаться в соответствии со следующими правилами:

– непомеченные ветви содержат все объекты, указанные в общей метке дуги после слияния (т.е. все объекты исходят из всех ветвей);

– помеченные перед слиянием ветви содержат все или некоторые объекты из перечисленных в общей метке после слияния (т.е. метка ветви ясно указывает, что содержит ветвь).

Например, *сырье и заготовки* как часть дуги *рабочий комплект* сходятся вместе с *принятым, но незаконченным заданием* для формирования главного входа в функциональный блок «Выполнить задание». *Сырье и заготовки* - это название, включающее и те, и другие объекты, поэтому дуга после соединения получает эту метку.

Дерево узлов

Дерево узлов это обзорная диаграмма, показывающая структуру всей модели (рис. 4). Обычно структура соответствует контекстному блоку, под вершинами выстраивается вся иерархия функциональных блоков модели.

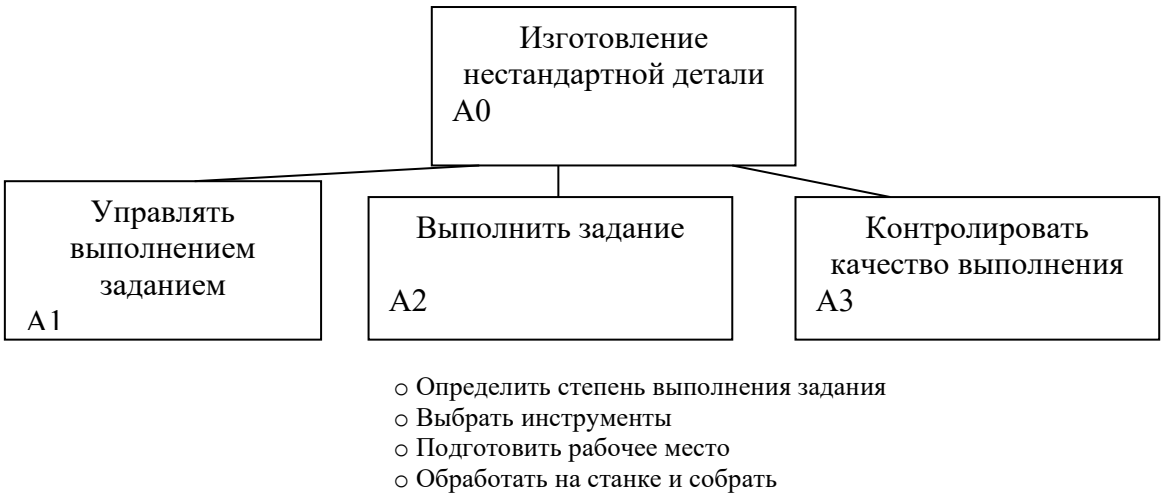

Рисунок 4. Дерево узлов

Требования к IDEF3 модели

Основой модели IDEF3 служит *сценарий* (*алгоритм*) процесса, который выделяет последовательность действий анализируемой системы, с одновременным описанием объектов, имеющих непосредственное отношение к процессу. Диаграмма является основной единицей описания в IDEF3.

IDEF3 модель органично дополняет модели IDEF0 и может являться описанием некоторого процесса анализируемой системы ПО. В этом случае IDEF3 модель быть получена на основе декомпозиции некоторой функции модели IDEF0 (например, *Подготовить рабочее место* рис. 5). IDEF3 модель может быть также использована как метод создания процессов (рис. 6). При разработке IDEF3 модели ПО в курсовом проекте может быть использован один из этих подходов **(по желанию студента).**

IDEF3 модель некоторого процесса ПО должна включать:

- *Цель модели;*
- *Точку зрения модели;*
- *Список работ* **(не менее 4-х работ)***;*
- *Описание работ;*
- *Список объектов* **(не менее 2-х объектов)***;*

– *Описание объектов*
– *Диаграмму* **(включающую не менее 3-х перекрестков)**

Диаграмма может быть выполнена без применения специального бланка. Работы и объекты, отображаемые на диаграмме, должны быть описаны.

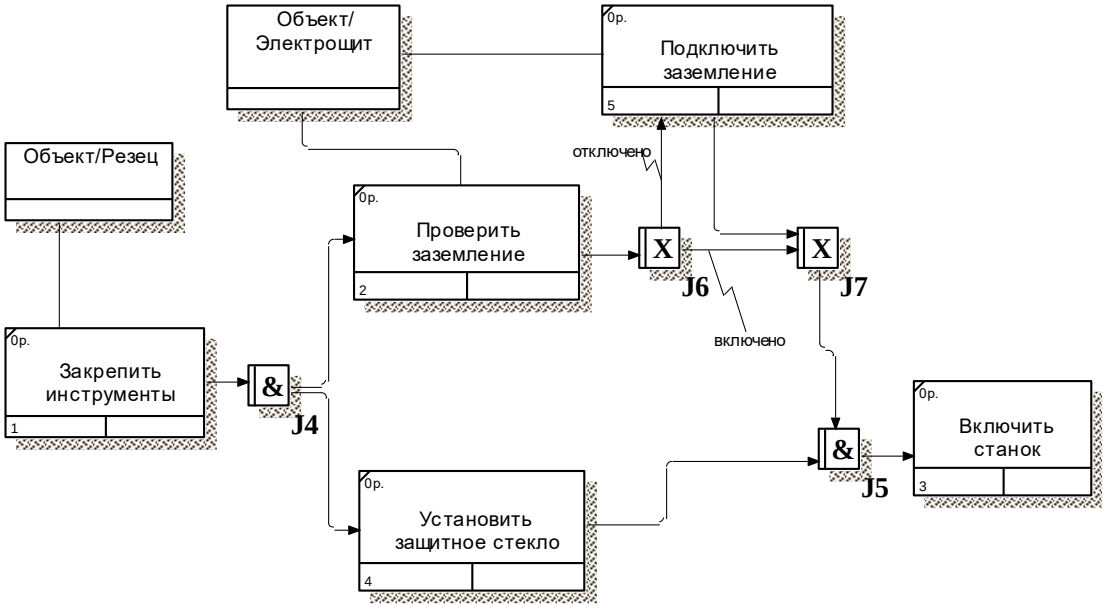

Рисунок 5. Декомпозиция (в нотации IDEF3) функции «Подготовить рабочее место»

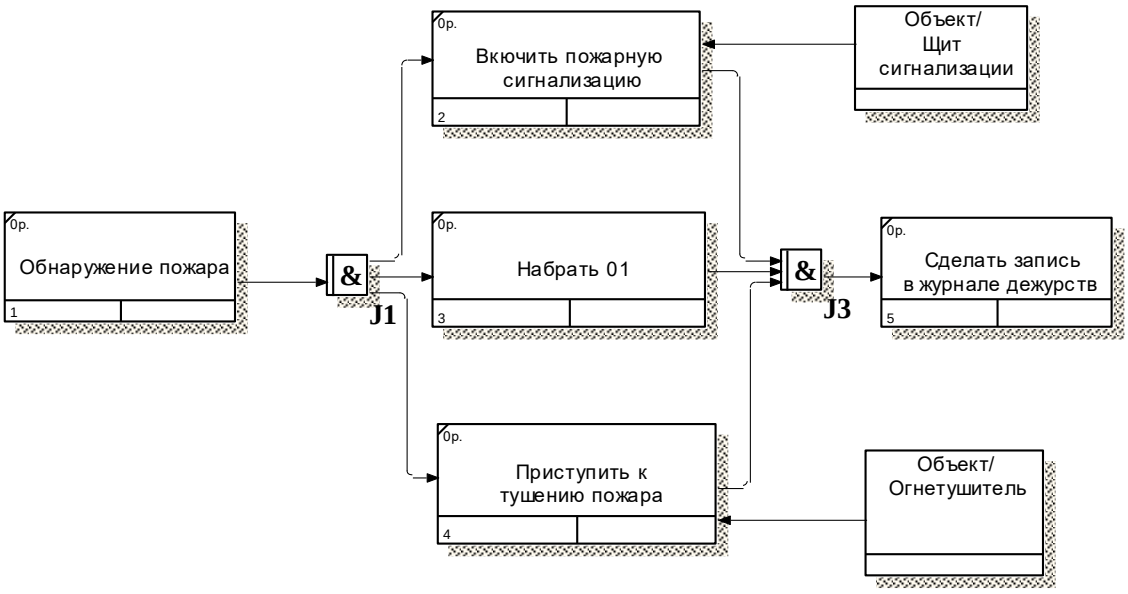

Рисунок 6. Моделирование процесса «Обнаружение и тушение пожара»

Краткая справка по синтаксису IDEF3 диаграмм*

Единицы работы (Unit of Work или UOW), также называемые работами, являются центральными компонентами модели. В IDEF3 работы изображаются прямоугольниками с прямыми углами (рис. 7) и 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 (например, «*Сборка изделия»*).

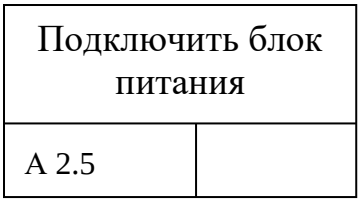

Рисунок 7. Изображение и нумерация действия в диаграмме IDEF3

В IDEF3 декомпозиция используется для детализации работ. Методология IDEF3 позволяет декомпозировать работу многократно, т. е. работа может иметь множество дочерних работ. Это позволяет в одной модели описать альтернативные потоки. Возможность множественной декомпозиции предъявляет дополнительные требования к нумерации работ. Так, номер работы A 3.2.5 состоит из номера родительской работы - 3, версии декомпозиции -2 и собственного номера работы на текущей диаграмме - 5

Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи:

✓ ——→ показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется;
✓ - - - - - - - → используется для изображения связей между единицами работ, а также между единицами работ и объектами ссылок. Задается аналитиком отдельно для каждого случая;
✓ ——→→ применяется для описания того факта, что объект используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ. Перекрестки используются для отображения логики

* см. также
Маклаков С.В. Создание информационных систем с AllFusion Modeling Suite. - М.: ДИАЛОГ – МИФИ, 2003.
Черемных С.В. и др. Моделирование и анализ систем. IDEF – технологии: Практикум. – М.: Финансы и статистика, 2003.

взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния и разветвления стрелок (таб. 1). Перекресток не может использоваться одновременно для слияния и для разветвления.

Таблица 1
Типы соединений

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
&	Асинхронное И	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
&	Синхронное И	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
O	Асинхронное ИЛИ	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
O	Синхронное И	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
X	Эксклюзивное ИЛИ	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J.

Диаграммы IDEF3 могут содержать указатели, которые предназначены для привлечения внимания читателя к каким-либо важным аспектам модели. Указатели изображаются на диаграмме в виде прямоугольника, похожего на изображение работы. Имя указателя включает его тип (рис. 8). Среди указателей различают типы: объект, ссылка, заметка, уточнение.

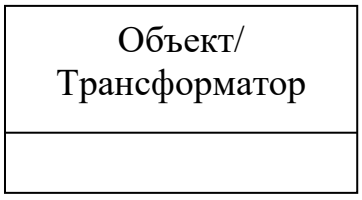

Рисунок 8. Указатель типа «Объект»

Требования к DFD модели

Диаграммы потоков данных (DFD) обеспечивают удобный способ описания передаваемой информации, как между частями системы, так и между системой и внешним миром. Поэтому они используются для создания моделей информационного обмена организации, например, документооборота. DFD модель органично дополняет модели IDEF0 и IDEF3 и является описанием информационного обмена некоторой подсистемы анализируемой системы ПО.

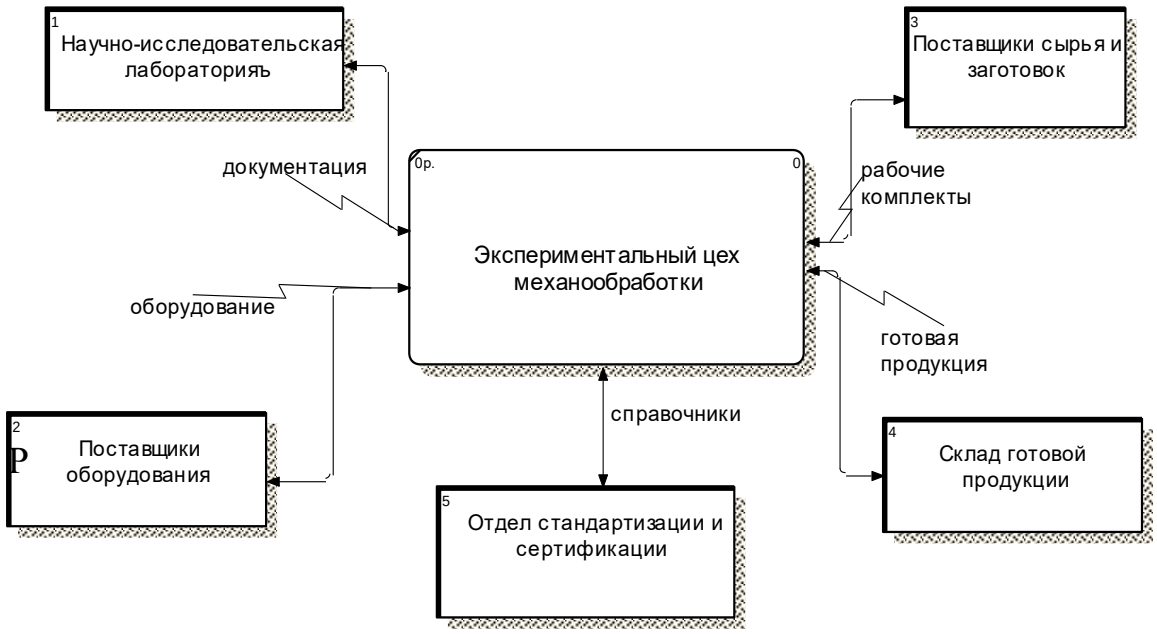

Рисунок 9. Контекстная диаграмма DFD

Таким образом, модель может быть получена на основе декомпозиции некоторой функции модели IDEF0 (например, *Контролировать качеством выполнения* рис. 10). При разработке в курсовом проекте DFD модели ПО это необходимо учесть.

DFD модель ПО должна включать:

- *Название модели;*
- *Список функций обработки информации (работы)* **(не менее 3-х);**
- *Описание функциональных блоков;*
- *Список данных;*
- *Список хранилищ данных* **(не менее 2-х)***;*
- *Список внешних сущностей* **(не менее 2-х)***;*
- *Диаграммы:*

1. Контекстная диаграмма (пример на рис. 9)
2. Диаграмма декомпозиции некоторой функции (пример на рис. 10).

Диаграммы могут быть выполнены без применения специального бланка. Функции, данные, сущности и хранилища, отображаемые на диаграммах, должны быть описаны.

Краткая справка по синтаксису DFD диаграмм*

Подобно IDEF0, DFD представляет модельную систему как сеть связанных между собой работ. DFD диаграмма содержит функциональные блоки (работы), внешние сущности, стрелки, хранилища данных.

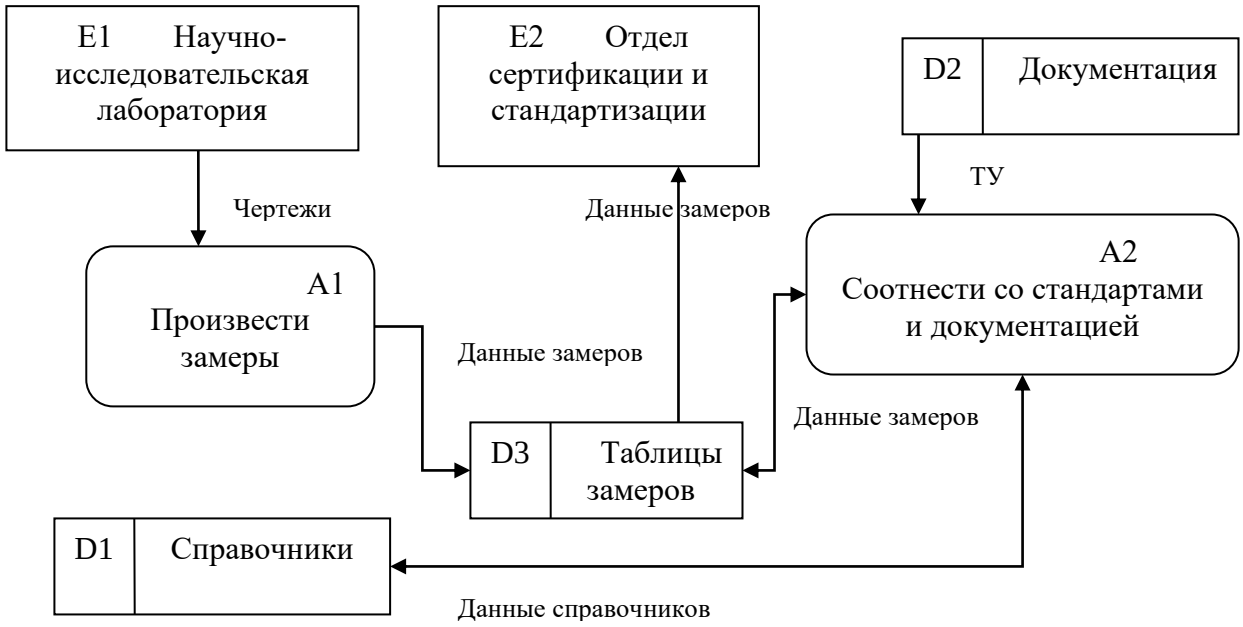

Рисунок 10. Диаграмма DFD для подсистемы «Контролировать качество выполнения»

Контекстная диаграмма, как правило, состоит из одного функционального блока и нескольких внешних сущностей. Функциональный блок при этом имеет имя совпадающее с именем всей системы.

В DFD работы представляют собой функции системы, преобразующие входы в выходы. Хотя работы изображаются прямоугольниками со скругленными углами, смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же как работы IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0.

* см. также
Маклаков С.В. Создание информационных систем с AllFusion Modeling Suite. - М.: ДИАЛОГ – МИФИ, 2003.
Черемных С.В. и др. Моделирование и анализ систем. IDEF – технологии: Практикум. – М.: Финансы и статистика, 2003.

Внешние сущности изображают входы в систему и/или выходы из системы. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок.

Стрелки описывают движение объектов из одной части системы в другую. 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, описывающих объекты в движении, хранилища данных изображают объекты в покое (рис. 11). В материальных системах хранилища данных изображаются там, где объекты ожидают обработки, например в очереди. В системах обработки информации хранилища данных являются механизмом, который позволяет сохранить данные для последующих процессов.

Рисунок 11. Изображение хранилища данных в DFD модели.

В DFD номер каждой работы может включать префикс, номер родительской работы (A) и номер объекта. Номер объекта – это уникальный номер работы на диаграмме. Например, работа может иметь номер A.12.4. Уникальный номер имеют хранилища данных и внешние сущности независимо от их расположения на диаграмме. Каждое хранилище данных имеет префикс D и уникальный номер, например D5. Каждая внешняя сущность имеет префикс E и уникальный номер, например E5.

Требования к инфологической модели

Инфологическая модель должна быть разработана на основе представленной в курсовом проекте DFD модели. Таким образом, это будет модель некоторой подсистемы рассматриваемой системы ПО. Рассмотрите хранилища данных DFD модели и опишите сущности, атрибуты и связи, соответствующие данным хранилищам (пример на рис.12).

Модель должна включать:

– не менее ***4*** *сущностей***;**

– среди *атрибутов сущностей* должны быть простые однозначные и многозначные атрибуты; а также атрибуты признаки и основания.

– среди с*вязей* должны присутствовать связи типа «один к многим»;

– сведения о *ключевых атрибутах* сущностей.

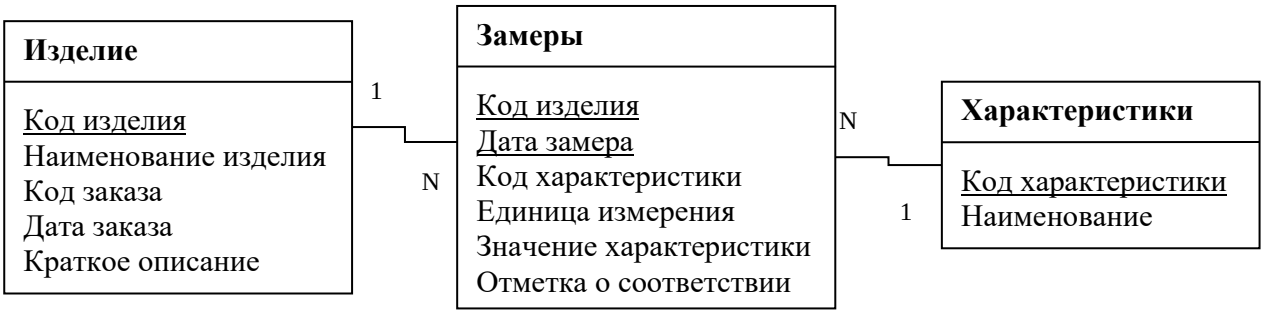

Рисунок 12. Инфологическая модель хранилища данных «Таблицы замеров» подсистемы «Контролировать качество выполнения» системы «Экспериментальный цех механообработки».

Модель может быть описана на языке ER-диаграмм, при этом допускается гибридная нотация с разъяснением обозначений. Атрибуты сущности должны удовлетворять требованиям *третьей нормальной* формы реляционной модели.

Краткая справка по процессу проведения нормализации*

1. Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения (значения в домене не являются ни списками, ни множествами простых или сложных значений);
2. Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый неключевой атрибут полностью зависит от первичного ключа (не должно быть зависимости от части ключа). Вторая нормальная форма имеет смысл только для сущностей, имеющих сложный первичный ключ.
3. Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой неключевой атрибут не зависит от другого неключевого атрибута (не должно быть взаимозависимости между неключевыми атрибутами).

---

* см. также

Базы данных: модели, разработка, реализация/ Под. ред. Карповой Т,С, - СПб.: Питер, 2004.

Глушаков С.В., Ломотько Д.В. Базы данных. – Харьков: Фолио; М.: ООО Издательство АСТ, 2002.

Требования к логической модели

Логическая структура реляционной базы данных должна быть разработана на основе инфологической модели, представленной в курсовом проекте. Каждая реляционная таблица должна иметь структуру, определяемую реквизитным составом одного из информационных объектов полученной инфологической модели. Логические связи таблиц должны соответствовать структурным связям между объектами (пример 1).

Логическая модель должна быть описана средствами языка SQL:

Создание базы данных

CREATE DATABASE <имя_базы_данных>

Создание таблицы

CREATE TABLE <имя_таблицы>
(<имя_столбца> <тип_столбца> [NOT NULL][UNIQUE | PRIMARY KEY]
[REFERENCES <имя_таблицы> [<имя_столбца>]] [, n] ...)

Создание индекса

CREATE [UNIQUE] INDEX <имя_индекса> ON <имя_таблицы> (<имя_столбца>,...)

Пример 1.

Создание базы данных «Данные замеров»

```
CREATE DATABASE D dan_zam
```

Создание таблицы «Изделие»

```
CREATE TABLE izdel
(i_kod INT NOT NULL PRIMARY KEY,
 i_name CHAR(20),
 i_kodzak INT NOT NULL,
 i_datzak SMALLDATETIME,
 i_krop  CHAR(40))
```

Создание таблицы «Характеристики»

```
CREATE TABLE charact
(c_kod INT NOT NULL PRIMARY KEY,
 c_name CHAR(30))
```

Создание таблицы «Замеры»

```
CREATE TABLE zamer
(i_kod INT NOT NULL REFERENCES izdel,
 z_datzam SMALLDATETIME  NOT NULL,
 c_kod  INT NOT NULL REFERENCES charact,
 z_ed CHAR(4),
 z_znach REAL,
 z_otm  CHAR(10),
 PRIMARY KEY (i_kod, z_datzam))
```

Создание индекса к таблице «Изделие»

CREATE INDEX inizdel ON izdel (i_datzak)

Требования к физической модели*

Выполнить описание базы данных в среде SQL Server. Разработать данные контрольного примера для описанной базы данных.

Внести в таблицы базы данных данные контрольного примера, воспользовавшись оператором:

INSERT INTO <имя_таблицы> [(<имя_столбца>,<имя_столбца>,...)]

VALUES (<значение>,<значение>,..)

или средствами утилиты Enterprise Manager (пример 2).

Пример 2.

Ввод данных в таблицу «Изделие» (обратите внимание на формат даты)

INSERT INTO izdel VALUES (100, 'Подшипник', 521, '12/05/2006', 'Тестирование для цеха №1')

Модель должна включать:

– *описание системы кодирования* для атрибутов, содержащих кодовые обозначения (например, атрибут *Код изделия* на рис. 12);

– *4-5 записей* для каждого отношения;

– *2-3 представления* для различных таблиц базы данных (пример 3);

– *процедуры без параметров и с параметрами* для изменения значений атрибутов одной из таблиц при выполнении некоторого условия (пример 4);

– *триггер* для одной из таблиц, контролирующий либо операцию изменения записей в таблице, либо добавление записей в таблицу, либо удаление записей из таблицы (пример 5);

– *описание запросов* к таблицам, следующего характера (пример 6):

1. запрос на выборку записей, удовлетворяющих некоторому условию с использованием логической операции проверки на вхождение в диапазон;
2. запрос на выборку записей, удовлетворяющих некоторому условию с использованием логической операции проверки на вхождение в множество;
3. запрос на выборку записей из таблицы, являющейся результатом слияния двух других таблиц по некоторому условию;
4. запрос с использованием агрегатных функций;
5. запрос на выборку записей с условием сортировки;
6. вложенный запрос на выборку записей с использованием предиката EXISTS.

---

* см. также

Базы данных: модели, разработка, реализация/ Под. ред. Карповой Т,С, - СПб.: Питер, 2004.

Глушаков С.В., Ломотько Д.В. Базы данных. – Харьков: Фолио; М.: ООО Издательство АСТ, 2002

– *пользовательское приложение*, реализующее экранный интерфейс с базой данных для ее просмотра и редактирования посредством ODBC (для разработки приложения могут быть использованы Delphi, Visual Basic, C++Builder, MS Access и др.).

– *отчетные формы,* сгенерированные на основании таблиц БД и реализованные либо в пользовательском приложении, либо с использованием технологии слияния текстовых документов с базой данных.

База данных и пользовательское приложение должны быть представлены на дискете. Представления, триггеры и процедуры, отчетные формы должны быть также описаны в тексте курсового проекта.

Пример 3.

Создание представления, которое содержит информацию об изделиях с датой заказа ранее 12/05/2006

```
CREATE VIEW oldizdel  AS SELECT i_kod, i_name, i_datzak FROM izdel
WHERE i_datzak <’12/05/2006’
```

Пример 4.

Создание процедуры, реализующей изменение в таблице «Замеры», реализующей увеличение значения характеристики в 100 раз

```
CREATE PROCEDURE new_edizm AS
UPDATE zamer
SET z_znach= z_znach*100
```

Запуск процедуры new_edizm

```
EXEC new_edizm
```

Создание процедуры с параметрами, увеличивающей в таблице «Замеры» значение характеристики в 100 раз при указании кода изделия, для которого необходимо выполнить данное увеличение

```
CREATE PROCEDURE new_edizm1 (@kod INT) AS
UPDATE zamer
SET z_znach= z_znach*100  WHERE i_kod=@kod
```

Запуск процедуры new_edizm1 *для изделия с кодом 120*

```
EXEC new_edizm1 @kod=120
```

Пример 5.

Создание триггера для обработки операции удаления зависимых записей из таблицы Изделия при удалении записей из таблицы Замеры.

```
CREATE TRIGGER udalen ON zamer
FOR DELETE
AS
IF @@ROWCOUNT=1
BEGIN
DECLARE @x INT
SELECT @x=d.i_kod FROM zamer a, deleted d WHERE a.i_kod=d.i_kod
```

```
IF EXISTS (SELECT * FROM izdel WHERE i_kod=@x)
DELETE FROM izdel WHERE i_kod=@x
END
```

Пример 6.

6.1. SELECT * FROM izdel WHERE i_kod BETWEEN 100 AND 200

SELECT z_datzam, z_otm FROM zamer WHERE z_datzam<’10/12/2005’ AND z_datzam>’08/10/2004’

6.2. SELECT * FROM izdel WHERE i_kod IN (100,200,300,400)

6.3. SELECT izdel.i_kod, izdel.i_datzak, zamer.z_datzam FROM izdel, zamer WHERE izdel.i_kod = zamer.i_kod

6.4. SELECT COUNT(*) AS ‘количество изделий’ FROM izdel

SELECT MIN(z_znach) AS ‘минимальное значение характеристики’ FROM zamer

6.5. SELECT * FROM izdel ORDER BY i_datzak ASC

6.6. *Вывести список изделий, для которых не проводились замеры*

SELECT i_name, i_krop FROM izdel WHERE NOT EXISTS (SELECT i_kod FROM zamer WHERE izdel.i_kod=zamer.i_kod)

Краткая справка по языку запросов SQL

Оператор SELECT – один из наиболее важных и самых распространенных операторов SQL. Он позволяет производить *выборки* (запросы) данных из таблиц и преобразовывать к нужному виду полученные результаты.

Оператор SELECT имеет следующий формат:

```
SELECT [ALL | DISTINCT ] {*|[имя_столбца
  [AS  новое_имя]]} [,...n]
  FROM  имя_таблицы [[AS] псевдоним] [,...n]
  [WHERE  <условие_поиска>]
  [GROUP BY  имя_столбца [,...n]]
  [HAVING <критерии выбора групп>]
  [ORDER BY имя_столбца [,...n]]
```

Обработка элементов оператора SELECT выполняется в следующей последовательности:

1. FROM – определяются имена используемых таблиц;
2. WHERE – выполняется *фильтрация строк* таблицы в соответствии с заданными условиями;
3. GROUP BY – образуются *группы строк*, имеющих одно и то же значение в указанном столбце;
4. HAVING – фильтруются группы строк таблицы в соответствии с указанным условием;
5. SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;

6. ORDER BY – определяется упорядоченность результатов выполнения операторов (сортировка).

Порядок предложений и фраз в операторе SELECT не может быть изменен. Только два предложения SELECT и FROM являются обязательными, все остальные могут быть опущены. SELECT – закрытая операция: *результат запроса* к таблице представляет собой другую таблицу.

С помощью WHERE-параметра пользователь определяет, какие блоки данных из приведенных в списке FROM таблиц появятся в *результате запроса*. За ключевым словом WHERE следует перечень *условий поиска*, определяющих те строки, которые должны быть выбраны при выполнении запроса. Существует пять основных типов *условий поиска* (или предикатов):

Сравнение: сравниваются результаты вычисления одного выражения с результатами вычисления другого (операторы *сравнения*: = – равенство; < – меньше; > – больше; <= – меньше или равно; >= – больше или равно; <> – не равно).

Диапазон: проверяется, попадает ли результат вычисления выражения в заданный диапазон значений (Оператор BETWEEN используется для поиска значения внутри некоторого интервала, определяемого своими минимальным и максимальным значениями. При этом указанные значения включаются в *условие поиска*, например, *WHERE Цена BETWEEN 100 And 150*).

Принадлежность множеству: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений (Оператор IN используется для *сравнения* некоторого значения со списком заданных значений, при этом проверяется, соответствует ли результат вычисления выражения одному из значений в предоставленном списке. NOT IN используется для отбора любых значений, кроме тех, которые указаны в представленном списке. Например, *WHERE Город IN (”Москва”, ”Самара”)).*

Соответствие шаблону: проверяется, отвечает ли некоторое строковое значение заданному шаблону.

Значение NULL: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение).

С помощью *итоговых (агрегатных) функций* в рамках SQL-запроса можно получить ряд обобщающих статистических сведений о множестве отобранных значений выходного набора. Итоговые функции могут использоваться только в списке предложения SELECT и в составе предложения HAVING. Во всех других случаях это недопустимо. Если список в предложении SELECT содержит итоговые функции, а в тексте запроса отсутствует фраза GROUP BY, обеспечивающая объединение данных в группы, то ни один из элементов списка предложения SELECT не может включать каких-либо ссылок на поля, за исключением ситуации, когда поля выступают в качестве аргументов итоговых функций.

Пользователю доступны следующие основные *итоговые функции*:

COUNT (Выражение) - определяет количество записей в выходном наборе SQL-запроса;

MIN/MAX (Выражение) - определяют наименьшее и наибольшее из множества значений в некотором поле запроса;

AVG (Выражение) - эта функция позволяет рассчитать среднее значение множества значений, хранящихся в определенном поле отобранных запросом записей. Оно является арифметическим средним значением, т.е. суммой значений, деленной на их количество.

SUM (Выражение) - вычисляет сумму множества значений, содержащихся в определенном поле отобранных запросом записей.

Чаще всего в качестве выражения выступают имена столбцов. Выражение может вычисляться и по значениям нескольких таблиц.

Все эти функции оперируют со значениями в единственном столбце таблицы или с арифметическим выражением и возвращают единственное значение. Функции COUNT, MIN и MAX применимы как к числовым, так и к нечисловым полям, тогда как функции SUM и AVG могут использоваться только в случае числовых полей, за исключением COUNT(*).

Представление - это предопределенный запрос, хранящийся в базе данных, который выглядит подобно обычной таблице и не требует для своего хранения дисковой памяти.

Создание и изменение *представлений:*

```
CREATE| ALTER VIEW имя_представления
    AS SELECT_оператор
```

Хранимые процедуры представляют собой группы связанных между собой операторов SQL, применение которых делает работу программиста более легкой и гибкой, поскольку выполнить хранимую процедуру часто оказывается гораздо проще, чем последовательность отдельных операторов SQL. Хранимые процедуры представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде.

Создание новой и изменение имеющейся *хранимой процедуры* осуществляется с помощью следующей упрощенной команды:

```
CREATE | ALTER  PROC[EDURE] имя_процедуры
   [{@имя_параметра тип_данных } ]
AS
  sql_оператор [...n]
```

Для передачи входных и выходных данных в создаваемой хранимой процедуре могут использоваться *параметры*, имена которых, как и имена локальных переменных, должны начинаться с символа @. В одной хранимой процедуре можно задать множество параметров, разделенных запятыми. В теле процедуры не должны применяться локальные переменные, чьи имена совпадают с именами параметров этой процедуры. Для определения типа данных, который будет иметь соответствующий параметр хранимой процедуры, годятся любые типы данных SQL, включая определенные пользователем.

Для выполнения хранимой процедуры используется команда:

EXEC имя_процедуры [@имя_параметра=]{значение| @имя_переменной}[,...n]

Существует несколько способов передачи данных между командами. Один из них – передача данных через локальные переменные. Прежде чем использовать какую-либо переменную, ее следует объявить. Объявление переменной выполняется командой DECLARE, имеющей следующий формат:

DECLARE {@имя_переменной тип_данных }
[,...n]

Значения переменной можно присвоить посредством команд SET и SELECT. С помощью команды SELECT переменной можно присвоить не только конкретное значение, но и результат вычисления выражения. Например, *DECLARE @a INT, SET @a=10, SELECT @k=MIN(i_kod)) FROM izdel*

Триггер – это откомпилированная SQL-процедура, исполнение которой обусловлено наступлением определенных событий внутри реляционной базы данных. Триггер представляет собой специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблицах, с которыми триггеры связаны. Каждый триггер привязывается к конкретной таблице.

В отличие от обычной подпрограммы, триггер выполняется неявно в каждом случае возникновения *триггерного события*, к тому же он не имеет аргументов.

Основной формат команды создания или изменения триггера:

CREATE | ALTER} TRIGGER имя_триггера
ON {имя_таблицы | имя_представления }
{ { FOR | AFTER | INSTEAD OF }
{ [DELETE] [,] [INSERT] [,] [UPDATE] }
AS
sql_оператор [...n] }

Триггерные события состоят из вставки (INSERT), удаления (DELETE) и обновления (UPDATE) строк в таблице.

Параметры AFTER и INSTEAD OF, определяют поведение триггеров:

AFTER - триггер выполняется после успешного выполнения вызвавших его команд.

INSTEAD OF - *Триггер* вызывается вместо выполнения команд.

При выполнении команд добавления, изменения и удаления записей сервер создает две специальные таблицы: *inserted и deleted*. В них содержатся списки строк, которые будут вставлены или удалены по завершении транзакции. Структура таблиц inserted и deleted идентична структуре таблиц, для которых определяется триггер. Для каждого триггера создается свой комплект таблиц inserted и deleted, поэтому никакой другой триггер не сможет получить к ним доступ. В зависимости от типа операции, вызвавшей выполнение триггера, содержимое таблиц inserted и deleted может быть разным:

команда INSERT – в таблице inserted содержатся все строки, которые пользователь пытается вставить в таблицу; в таблице deleted не будет ни одной

строки; после завершения триггера все строки из таблицы inserted переместятся в исходную таблицу;

команда DELETE – в таблице deleted будут содержаться все строки, которые пользователь попытается удалить; триггер может проверить каждую строку и определить, разрешено ли ее удаление; в таблице inserted не окажется ни одной строки;

команда UPDATE – при ее выполнении в таблице deleted находятся старые значения строк, которые будут удалены при успешном завершении триггера. Новые значения строк содержатся в таблице inserted. Эти строки добавятся в исходную таблицу после успешного выполнения триггера.

Для получения информации о количестве строк, которое будет изменено при успешном завершении триггера, можно использовать функцию @@ROWCOUNT; она возвращает количество строк, обработанных последней командой. Следует подчеркнуть, что триггер запускается не при попытке изменить конкретную строку, а в момент выполнения команды изменения. Одна такая команда воздействует на множество строк, поэтому триггер должен обрабатывать все эти строки.

Рекомендуемые предметные области для проектирования

1. *Производство продукции (завод, фирма, отдел):*
   - бухгалтерия;
   - отдел сбыта;
   - отдел снабжения;
   - плановый отдел;
   - цех;
   - лаборатория;
   - вспомогательные службы;
   - вычислительный центр
- *Производство услуг:*
- больница;
- библиотека;
- бюро ремонта;
- учебное заведение;
- агентство недвижимости;
- консалтинговая компания;
- магазин или торговый комплекс;
- страховая компания;
- спортивный клуб;
- гостиница и т.п.

Рекомендуемая литература

Основная литература:

1. Проектирование информационных систем : учебник и практикум для вузов / под общей редакцией Д. В. Чистова. — Москва : Издательство Юрайт, 2021. — 258 с. — (Высшее образование). — ISBN 978-5-534-00492-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/469199

2. Гутгарц, Р. Д. Проектирование автоматизированных систем обработки информации и управления : учебное пособие для вузов / Р. Д. Гутгарц. — Москва : Издательство Юрайт, 2021. — 304 с. — (Высшее образование). — ISBN 978-5-534-07961-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/474654

3. Григорьев, М. В. Проектирование информационных систем : учебное пособие для вузов / М. В. Григорьев, И. И. Григорьева. — Москва : Издательство Юрайт, 2021. — 318 с. — (Высшее образование). — ISBN 978-5-534-01305-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/470711

Дополнительная литература:

1. Грекул, В. И. Проектирование информационных систем : учебник и практикум для вузов / В. И. Грекул, Н. Л. Коровкина, Г. А. Левочкина. — Москва : Издательство Юрайт, 2021. — 385 с. — (Высшее образование). — ISBN 978-5-9916-8764-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/469757

2. Шишмарёв, В. Ю. Организация и планирование автоматизированных производств : учебник для вузов / В. Ю. Шишмарёв. — 2-е изд. — Москва : Издательство Юрайт, 2021. — 318 с. — (Высшее образование). — ISBN 978-5-534-11451-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/475850

Периодика:

Научный периодический журнал «Вестник Южно-Уральского государственного университета. Серия «Компьютерные технологии, управление, радиоэлектроника» : Научный рецензируемый журнал. https://vestnik.susu.ru/ctcr - Текст : электронный.

Пример оформления титульного листа

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ЧЕБОКСАРСКИЙ ИНСТИТУТ (ФИЛИАЛ) МОСКОВСКОГО ПОЛИТЕХНИЧЕСКОГО УНИВЕРСИТЕТА

Кафедра информационных технологий, электроэнергетики и систем управления

Курсовая работа
По дисциплине: Проектирование автоматизированных систем
Тема КР: «»_____»

вариант _____

Выпол
нил(а):студента группы
27.03.04-2д-1Иванов
Иван Иванович
учебный шифр **1811111**

Проверил(а):
ст. преподаватель каф. ИТЭСУ:
Данилова Н.Е.

Чебоксары 2022