МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

имени академика М.Д. Миллионщикова

Кафедра «Информационные технологии»

Х.К.Алиева

Методические рекомендации к лабораторным работам по дисциплине «Моделирование информационных процессов и систем»

Направление подготовки

09.03.02 Информационные системы и технологии

Направленность (профиль)

«Информационные системы и технологии»

«Информационные технологии в образовании»

«Информационные технологии в дизайне»

Квалификация

бакалавр

Грозный 20__

Составители:

Ассистент кафедры
«Информационные технологии» Алиева Х.К.

Рецензент:

Э.Д. Алисултанова, доктор педагогических наук, кандидат физико-математических наук, профессор, директор Института прикладных информационных технологий, заведующая кафедрой «Информатика и вычислительная техника»

Методические указания предназначены для бакалавров по направлению подготовки 09.03.02 Информационные системы и технологии института прикладных информационных технологий.

Методические рекомендации рассмотрены и утверждены на заседании кафедры «Информационные технологии»: Протокол №__ от _____.20__г.

Рекомендовано к изданию редакционно-издательским советом ГГНТУ

Содержание

Введение

Задача оптимизации бизнес-процессов может решаться по-разному. Обычно это некоторая методика, которая позволяет, используя существующий бизнес-процесс, создать его модель и изменять её в зависимости от поставленных целей, внедряя её затем на практике. Вместе бизнес-процесс, его модель и участники составляют информационную систему.

В настоящее время существует несколько классов информационных систем для решения задач моделирования и оптимизации бизнес-процессов. Перечисленные системы будут рассмотрены в процессе курса:

– IDEF0 (ICAM Definition 0) – методология функционального моделирования. Применяется для описания рабочих процессов (Work Flow). Разработана на основе SADT в рамках программы ICAM (Integrated Computer Aided Manufacturing). В силу схожести методик SADT и IDEF0 их часто не разделяют.

– DFD (data flow diagrams) – метод структурированного анализа и проектирования. Это традиционное визуальное представление информационных потоков внутри системы. Диаграмма потоков данных (DFD) широко используется для анализа и проектирования программного обеспечения.

– IDEF3 (Integrated DEFinition for Process Description Capture Method) – методология моделирования и стандарт документирования процессов, происходящих в системе. Широко применяется при разработке информационных систем.

– UML (Unified Modeling Language) – язык визуального моделирования, основанный на объектно-ориентированном подходе. UML включает в себя двенадцать типов диаграмм, которые позволяют описать статическую структуру системы и ее динамическое поведение.

Лабораторная работа №1

Тема: Функциональная модель предметной области (методология IDEF0)

Цель: Научиться строить функциональную модель в нотации IDEF0.

Задание: Построить модель предметной области, согласно выбранного варианта с помощью контекстной диаграммы и диаграмм декомпозиции.

Теоретическая часть:

Методология IDEF0 (Integrated DEFinition) – это совокупность методов, правил и процедур, предназначенных для построения функциональной модели предметной области. Функциональная модель IDEF0 отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями.

Данная методология применяется при создании новых систем для определения требований и функций и затем для разработки системы, удовлетворяющей требованиям и реализующей функции. Для действующих систем эта методология может использоваться для анализа функций, выполняемых системой, а также для наглядного представления «механизмов», посредством которых эти функции осуществляются. Основной сферой применения методологии IDEF0 является предпроектное обследование и анализ системы.

Результатом методологии IDEF0 является модель.

Модель – это описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы.

Модель состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга.

Диаграммы – главные компоненты модели, состоящие из блоков и дуг. Блоки (работы) изображают функции моделируемой системы. Дуги (стрелки) связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними.

Функциональные блоки (работы) означают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Имя работы должно быть выражено глаголом в неопределенной форме и не должно повторяться во всей модели.

IDEFO требует, чтобы в диаграмме было от 3 до 6 блоков. Эти ограничения поддерживают сложность диаграмм и модели на уровне, доступном для чтения, понимания и использования.

Дуги представляют собой некую информацию и именуются существительными. Место соединения дуги с функциональным блоком определяет тип интерфейса. Функциональный блок и интерфейсные дуги представлены на рисунке 1.

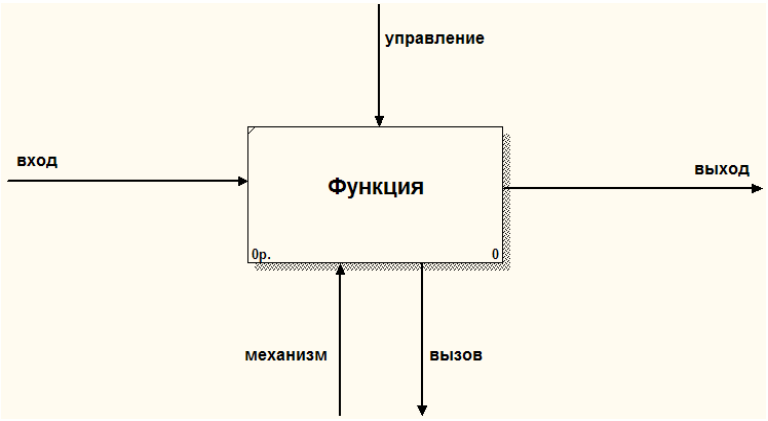


Рисунок 1 – Функциональный блок и интерфейсные дуги

В IDEF0 различают 5 типов стрелок:

1. **Вход (Input)** – материал или информация, которые используются или преобразуется функцией для получения результата (выхода). Допускается, что функция может не иметь ни одной стрелки входа. Зачастую сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить то, перерабатываются/изменяются ли данные в функции или нет. Если изменяются, то, скорее всего это вход, если нет – управление.
2. **Управление (Control)** – правила, стратегии, процедуры или стандарты, которыми руководствуется функция. Управление влияет на функцию, но не преобразуется функцией.
3. **Выход (Output)** – материал или информация, в которые преобразуются входы после выполнения функции. Функция без результата не имеет смысла и не должна моделироваться.
4. **Механизм (Mechanism)** – ресурсы, которые выполняют функцию, например, сотрудники предприятия, устройства и т.д.
5. **Вызов (Call)** – специальная стрелка, указывающая на другую модель функция. Стрелка вызова используется для указания того, что некоторая функция выполняется за пределами моделируемой системы. В BPwin стрелки вызова используются в механизме слияния и разделения моделей.

Декомпозиция функций

Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели. Декомпозиция

позволяет представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Сначала система моделируется как единое целое – один функциональный блок с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма называется контекстной.

В процессе декомпозиции функциональный блок подвергается детализации на другой диаграмме. Функциональные блоки диаграммы второго уровня (диаграммы декомпозиции или дочерние диаграммы) отображают главные подфункции функционального блока контекстной диаграммы и называются дочерними блоками. В свою очередь, функциональный блок-предок называется родительским блоком, а диаграмма, к которой он принадлежит – родительской диаграммой.

Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. При этом все интерфейсные дуги, входящие в функциональный блок или исходящие из него, фиксируются на дочерней диаграмме. Таким образом, достигается структурная целостность IDEF0-модели.

Следует обратить внимание на взаимосвязь нумерации функциональных блоков и диаграмм: каждый блок имеет свой уникальный порядковый номер на диаграмме (цифра в правом нижнем углу прямоугольника), а обозначение под правым углом указывает на номер дочерней для этого блока диаграммы. Отсутствие такого обозначения говорит о том, что декомпозиции для данного блока не существует.

Внутренние стрелки

Для связи работ между собой используются внутренние дуги, т.е. стрелки, которые не касаются границы диаграммы, начинаются у одной и заканчиваются у другой работы. В методологии IDEFO требуется только пять типов взаимодействий между работами для описания их отношений: управление, вход, обратная связь по управлению, обратная связь по входу, выход-механизм.

Связь по входу – стрелка выхода вышестоящей работы (далее - просто выход) направляется на вход нижестоящей (рис.2).

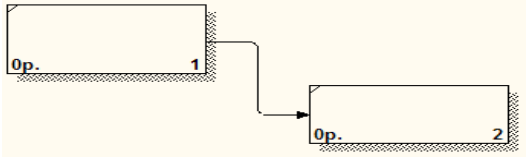


Рисунок 2 – Связь по входу

Связь по управлению – выход вышестоящей работы направляется на управление нижестоящей. Связь по входу показывает доминирование

вышестоящей работы. Данные или объекты выхода вышестоящей работы не меняются в нижестоящей (рис.3).

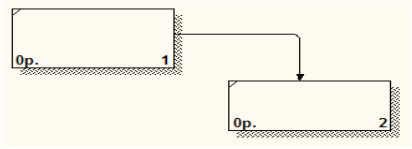


Рисунок 3 – Связь по управлению

Обратная связь по входу – выход нижестоящей работы направляется на вход вышестоящей. Такая связь, как правило, используется для описания циклов (рис.4).

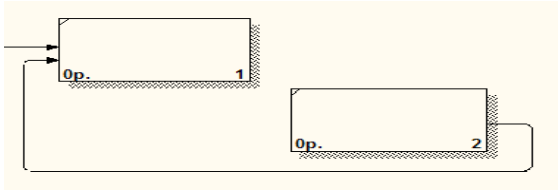


Рисунок 4 – Обратная связь по входу

Обратная связь по управлению – выход нижестоящей работы направляется на управление вышестоящей. Обратная связь по управлению часто свидетельствует об эффективности бизнес-процесса (рис.5).

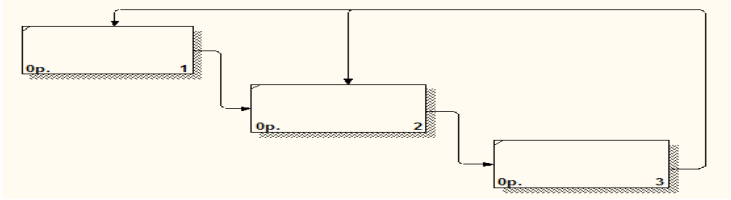


Рисунок 5 – Обратная связь по управлению

Связь выход-механизм – выход одной работы направляется на механизм другой. Эта взаимосвязь используется реже остальных и показывает, что одна работа подготавливает ресурсы, необходимые для проведения другой работы (рис.6).

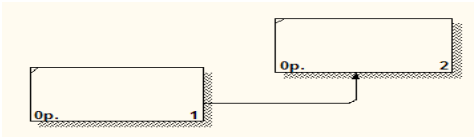


Рисунок 6 – Связь выход-механизм

Слияние и разветвление стрелок

Стрелки могут разветвляться и соединяться различными способами. Вся стрелка или ее часть может выходить из одного или нескольких блоков и заканчиваться в одном или нескольких блоках.

Разветвление дуг, изображаемое в виде расходящихся линий, означает, что все содержимое дуг или его часть может появиться в каждом ответвлении. 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;
- ветви, помеченные после точки разветвления, содержат все объекты или их часть, указанные в метке дуги перед разветвлением.

Слияния дуг в IDEFO, 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;
- помеченные перед слиянием ветви содержат все или некоторые объекты из перечисленных в общей метке после слияния.

Тоннелирование стрелок

Часто отдельные интерфейсные дуги не стоит рассматривать в дочерних диаграммах ниже или выше определенного уровня. Это будет только перегружать их и делать сложными для восприятия. Также бывает необходимо избавиться от отдельных «концептуальных» интерфейсных дуг и не детализировать их глубже некоторого уровня.

Для решения подобных задач в IDEF0 предусмотрено понятие тоннелированных стрелок (рис. 7). Символ «тоннеля» – две квадратные скобки вокруг начала интерфейсной дуги – обозначает, что дуга не была унаследована от функционального родительского блока и появилась только на этой диаграмме. «Тоннель» вокруг конца (стрелки) интерфейсной дуги в непосредственной близи от блока-приемника означает, что в дочерней по отношению к данному блоку диаграмме эта дуга отображаться и рассматриваться не будет.

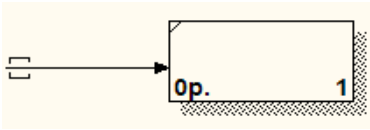


Рисунок 7 – Тоннелированная стрелка

Как правило, отдельные объекты и соответствующие им интерфейсные дуги «убираются» на промежуточных уровнях иерархии. В этом случае они сначала «погружаются в тоннель», а затем «возвращаются из тоннеля».

Рекомендации по рисованию диаграмм

В IDEF0 существует соглашения по рисованию диаграмм, которые призваны облегчить чтение и экспертизу модели. Некоторые из этих правил

BPwin поддерживает автоматически, выполнение других следует обеспечить вручную:

- прямоугольники работ должны располагаться по диагонали с левого верхнего в правый нижний угол;
- следует максимально увеличивать расстояние между входящими или выходящими стрелками на одной грани работы;
- следует максимально увеличивать расстояние между работами, поворотами и пересечениями стрелок;
- если две стрелки проходят параллельно (начинаются из одной и той же грани и заканчиваются на одной и той же грани другой работы), то по возможности следует их объединить и назвать единым термином;
- обратные связи по входу рисуются «нижней» петлей, обратная связь по управлению – «верхней»;
- циклические обратные связи следует рисовать только в случае крайней необходимости, когда подчеркивают значение повторно используемого объекта. Принято изображать такие связи на диаграмме декомпозиции;
- следует минимизировать число пересечений, петель и поворотов стрелок;
- если нужно изобразить связь по входу, необходимо избегать «нависания» работ друг над другом. В этом случае BPwin изображает связи по входу в виде петли, что затрудняет чтение диаграмм.

Модели AS-IS и TO-BE

Обычно сначала строится модель существующей организации работы – AS-IS (как есть). Анализ функциональной модели позволяет определить:

- наиболее слабые места;
- преимущества новых бизнес-процессов;
- глубину изменений, которым подвергнется существующая структура организации бизнеса.

Признаками неэффективной работы деятельности могут быть:

- бесполезные, неуправляемые и дублирующиеся работы;
- неэффективный документооборот;
- отсутствие обратных связей по управлению;
- отсутствие обратных связей по входу.

Найденные в модели AS-IS недостатки можно исправить при создании модели TO-BE (как будет) – модели новой организации бизнес-процессов. Модель TO-BE нужна для анализа альтернативных путей выполнения работы и документирования того, как компания будет делать бизнес в будущем.

Алгоритм создания модели:

Запустите ERwin (приложение Process Modeler). В диалоговом окне выберите позицию Create model, введите имя модели и тип IDEF0. Нажмите ОК (рис.1).

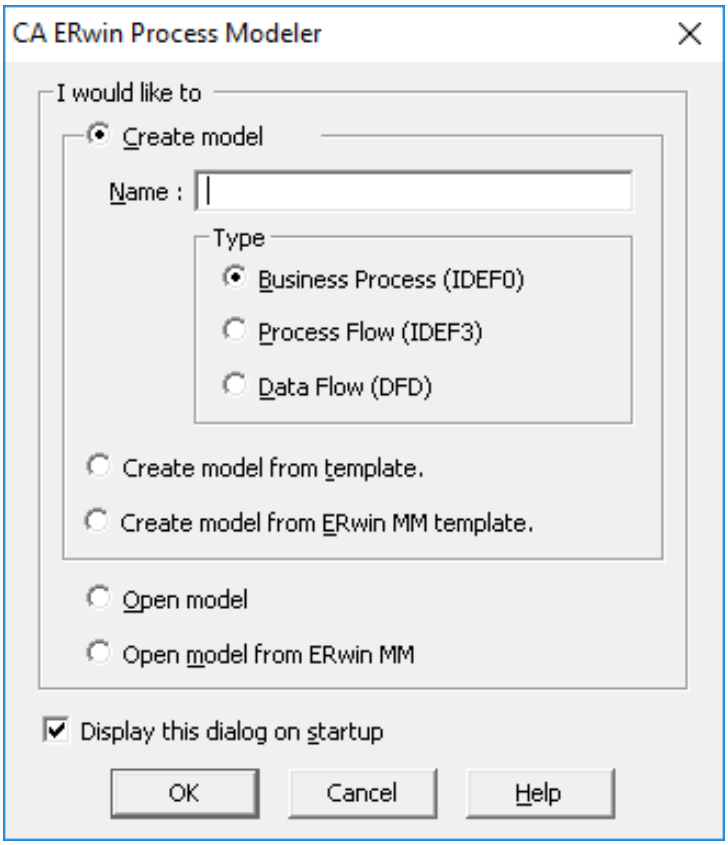


Рис.1.

Появится окно Properties for New Models. Во вкладке Author введите фамилию и инициалы автора. Остальные вкладки используются для определения настроек проекта (рис.2).

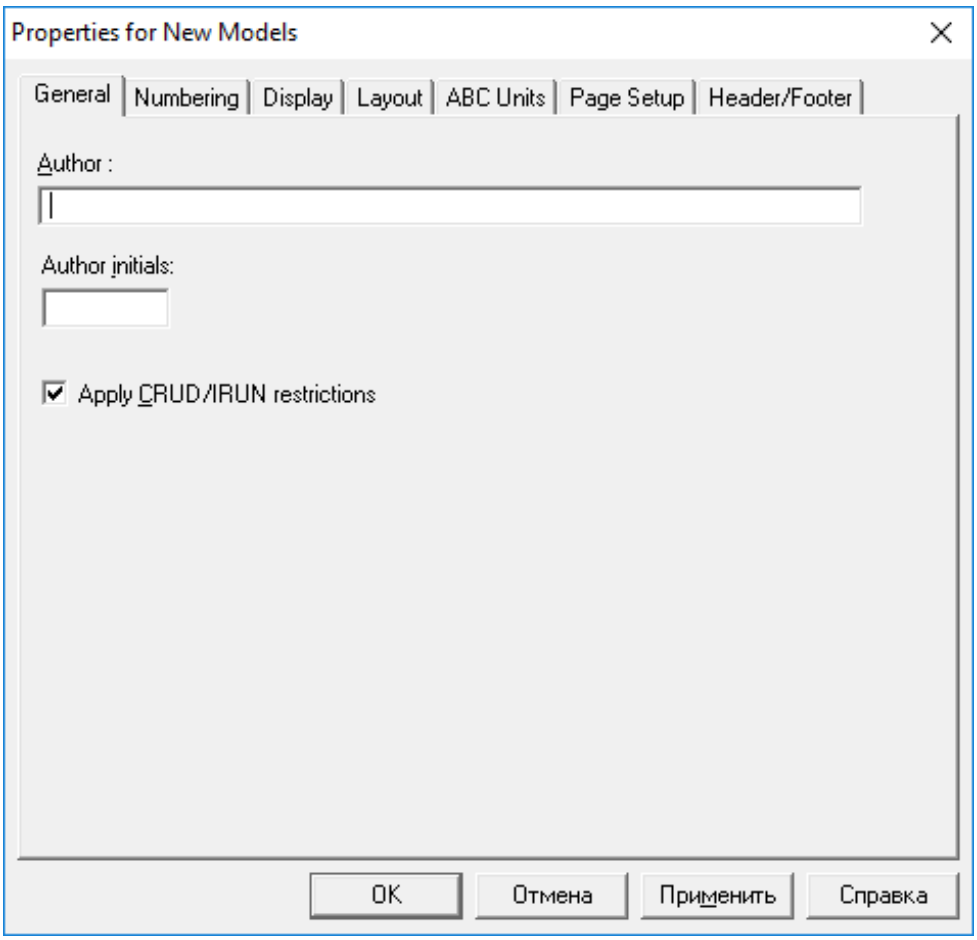


Рис.2.

В итоге, автоматически создается область для контекстной диаграммы с одним функциональным блоком.

Обратите внимание на панель инструментов IDEF0 со следующими кнопками (рис.3):

Рис.3.

 – кнопка для добавления функционального блока на диаграмму.

 – проведение новой связи.

 – инструмент редактирования объектов.

 – ссылка на пояснение стрелки

 – внесение текста в поле диаграммы

 – перемещение по моделям с их описанием

 – переход между стандартной диаграммой, деревом узлом и FEO

 – декомпозиция диаграммы нижнего уровня

 – декомпозиция диаграммы верхнего уровня

Для смены шрифта, выберите в верхнем меню «Model» и перейдите к пункту «Default Fonts». Далее выбирайте каждый пункт из списка по очередности, чтобы поменять шрифт (рис.4).

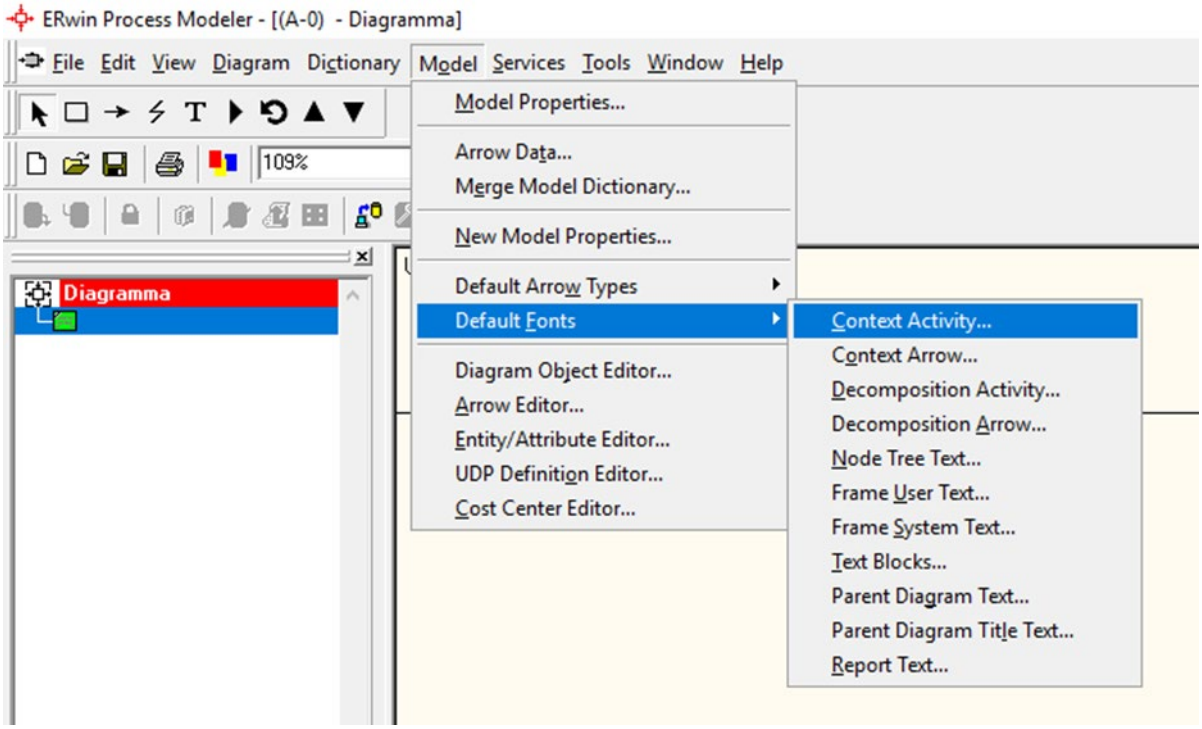


Рис.4.

В открывшемся окне необходимо выбрать Script – кириллический и указать шрифт (рис.5).

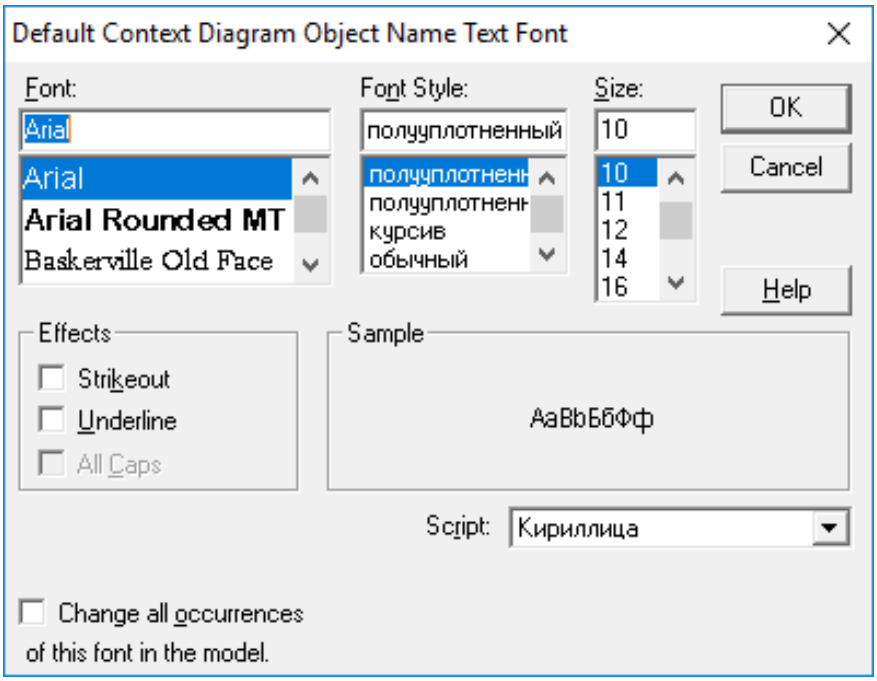


Рис.5.

Затем можно переходить к созданию контекстной диаграммы (рис.6).

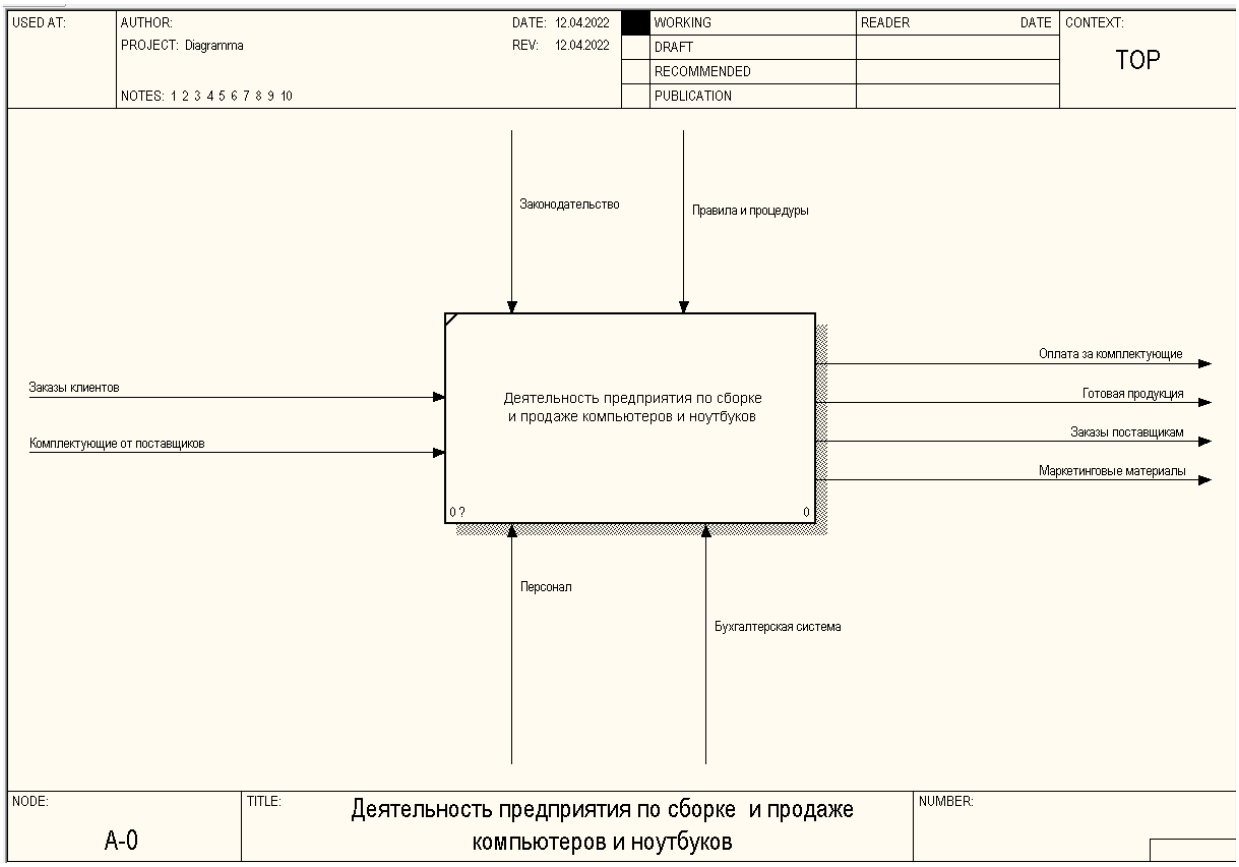


Рис.6.

На панели инструментов IDEF0 нажмите кнопку декомпозиции диаграммы нижнего уровня ▼. В появившемся диалоге Activity Box Count выберите методологию IDEF0 и количество блоков на диаграмме нижнего уровня – 4 (рис.7).

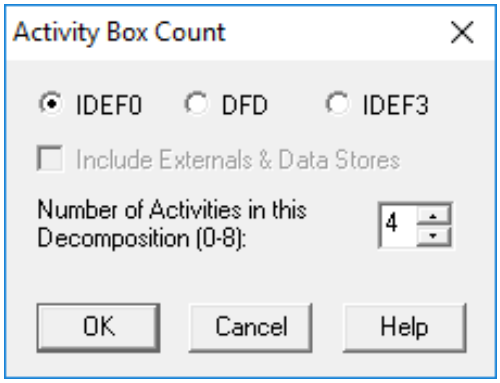


Рис.7.

Автоматически будет создана диаграмма декомпозиции. При декомпозиции стрелки контекстной диаграммы автоматически появились на диаграмме декомпозиции. Но они при этом не касаются работ. Такие стрелки называются несвязанными и считаются как синтаксическая ошибка.

Для связывания стрелок входа, управления и механизма необходимо перейдите в режим редактирования стрелок, щелкните по наконечнику

стрелки и щелкните по соответствующему сегменту работы. Для связывания выхода необходимо также перейти в режим редактирования стрелок, щелкнуть по сегменту выхода и затем по стрелке.

В результате выполненной декомпозиции бизнес-процесса получилась функциональная модель с детализацией до 1 уровня (рис.8).

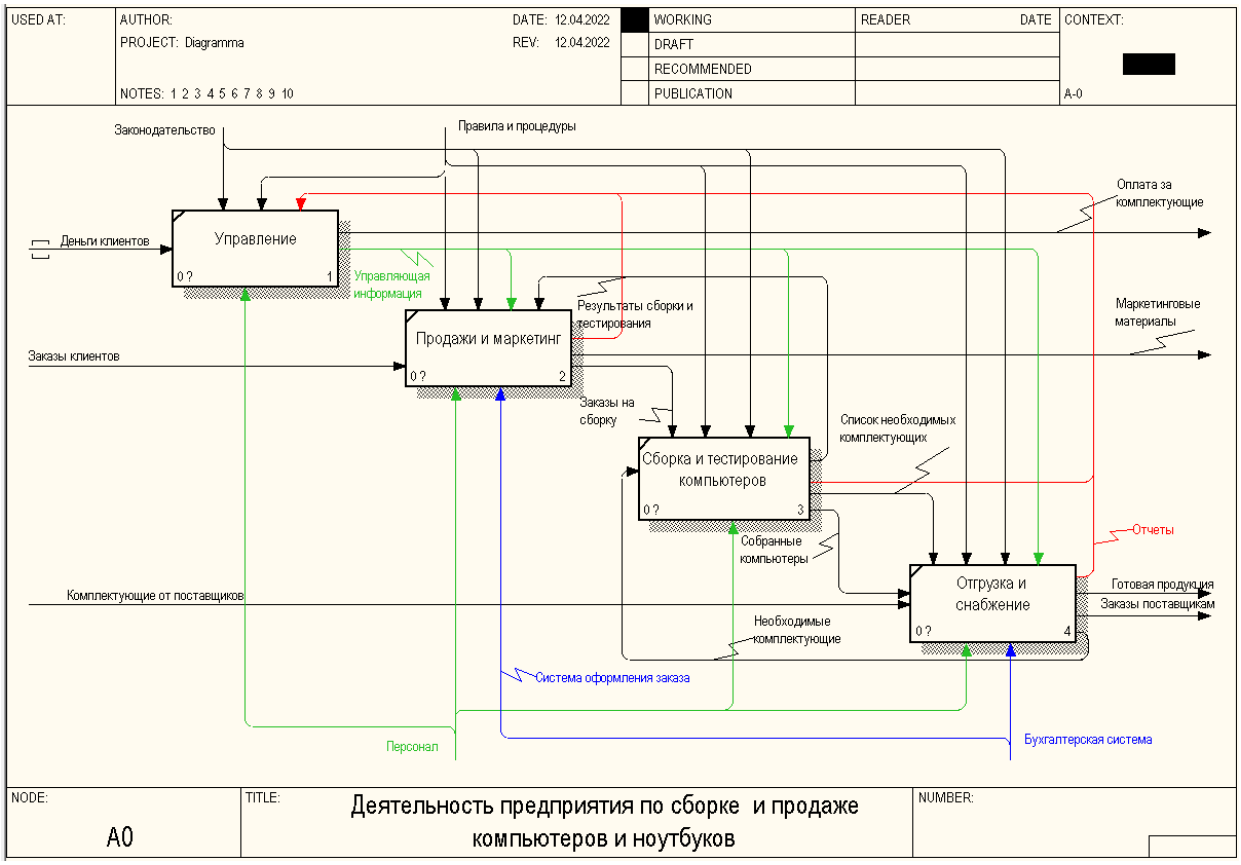


Рис.8.

Для дальнейшей детализации выбирают нужный блок и вновь повторяют процесс декомпозиции 2 уровня.

Порядок выполнения задания:

1. Выберите вариант темы.
2. Изучите предметную область.
3. Создайте новую модель.
4. Разработайте контекстную страницу модели.
5. Обдумайте, на какие функции может быть разложена главная функция системы, обозначенная Вами в функциональном блоке на контекстной странице модели. Число этих функций должно быть от 3 до 4.
6. Создайте диаграмму декомпозиции первого уровня. При создании диаграммы выберите в диалоговом окне нотацию диаграммы (IDEF0) и укажите, сколько функциональных блоков вы планируете разместить на диаграмме.

7. На диаграмме декомпозиции впишите названия выделенных функций в функциональные блоки. Помните о том, что функциональные блоки на диагонали должны быть расположены в порядке убывания их значимости или в соответствии с последовательностью выполнения работ.

8. Соедините интерфейсные дуги, которые мигрировали с диаграммы верхнего уровня на созданную диаграмму декомпозиции в виде стрелок, с функциональными блоками в соответствии с их назначением.

9. Если в этом есть необходимость, сделайте разветвления дуг. Помните о том, что Вы можете оставить единое название для всех веток. В этом случае название располагается до разветвления стрелки. В случае, если ветки обозначают разные объекты, подпишите каждую ветку.

10. Создайте внутренние дуги, связывающие функциональные блоки между собой. Помните, что каждый функциональный блок обязательно должен иметь дуги Управления и Выхода. Дуги Механизма и Входа могут отсутствовать. Именуйте каждую дугу.

11. По описанной выше технологии создайте диаграммы декомпозиции (от 3 до 4 функций) для тех функциональных блоков, прояснить содержание которых требуется по логике модели.

Контрольные вопросы:

1. Дайте определение методологии IDEF0;
2. Дайте определение диаграммам;
3. Что собой представляют функциональные блоки и дуги?
4. Какие виды стрелок различают в IDEF0?
5. Как определить являются ли данные входом или управлением?
6. В чем смысл декомпозиции функций?
7. Как нумеруются функциональные блоки?
8. Какие внутренние связи различают?
9. Что означает тонелирование стрелок и как выглядит?
10. Для чего используются модели AS-IS и TO-BE?

Лабораторная работа №2

Тема: Диаграмма потоков данных (DFD)

Цель: Научиться строить диаграмму потоков данных DFD.

Задание: Построить контекстную диаграмму и диаграмму декомпозиции в стандарте DFD.

Теоретическая часть:

Диаграммы потоков данных (Data Flow Diagrams - DFD) используются для описания движения документов и обработки информации как дополнение к IDEF0. В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, стрелки в DFD показывают лишь то, как объекты (включая данные) движутся от одной работы к другой. DFD отражает функциональные зависимости значений, вычисляемых в системе, включая входные значения, выходные значения и внутренние хранилища данных. DFD - это граф, на котором показано движение значений данных от их источников через преобразующие их процессы к их потребителям в других объектах.

DFD содержит процессы, которые преобразуют данные, потоки данных, которые переносят данные, активные объекты, которые производят и потребляют данные, и хранилища данных, которые пассивно хранят данные.

Диаграмма потоков данных содержит:

- процессы, которые преобразуют данные;
- потоки данных, переносящие данные;
- активные объекты, которые производят и потребляют данные;
- хранилища данных, которые пассивно хранят данные.

Источники информации - внешние сущности - порождают информационные потоки данных, переносящие информацию к процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам, хранилищам данных или внешним сущностям - потребителям информации.

Процесс (работа) – это преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом (рис.1). Каждый процесс имеет номер для его идентификации и имя. Имя начинается с глагола в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже. Работа имеет входы и выходы, но не поддерживает управление и механизмы, как IDEF0.

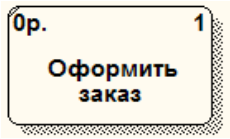


Рисунок 1 – Процесс (работа)

Внешняя сущность – это материальный предмет или физическое лицо, являющееся источником или приемником информации, например, заказчики, клиенты, бухгалтерия (рис. 2). Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой ИС. Внешняя сущность имеет номер для ее идентификации и имя. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах.

В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой ИС, если это необходимо, или, наоборот, часть процессов ИС может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

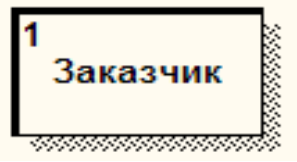


Рисунок 2 – Внешняя сущность

Поток данных (стрелка) – это информация, передаваемая через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т.д.

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока (рис. 3). Каждый поток данных имеет имя, отражающее его содержание. Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы.

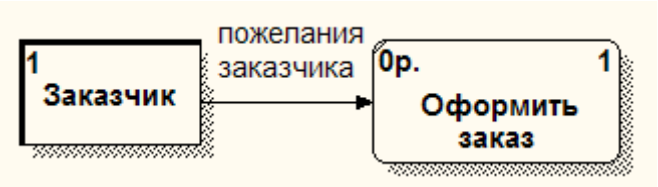


Рисунок 3 – Поток данных

Хранилище данных – это абстрактное устройство для хранения информации, которую можно в любой момент поместить в него и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми (рис. 4). Хранилище данных может быть реализовано физически в виде микрофиши, ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т.д.

Каждое хранилище данных имеет номер для его идентификации и имя. В случае, когда поток данных входит в хранилище или выходит из него и его

структура соответствует структуре хранилища, он должен иметь то же самое имя, которое нет необходимости отражать на диаграмме.

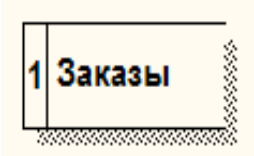


Рисунок 4 – Хранилище данных

Кроме основных элементов, в состав DFD входят словари данных и миниспецификации.

Словари данных являются каталогами всех элементов данных, присутствующих в DFD, включая потоки данных, хранилища и процессы, а также все их атрибуты. Миниспецификации обработки – описывают DFD процессы нижнего уровня. Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами: множество всех миниспецификаций является полной спецификацией системы.

Диаграммы потоков данных строятся в виде иерархии. Сначала строится контекстная диаграмма. При проектировании относительно простых систем строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы. Перед построением контекстной DFD необходимо проанализировать внешние события (внешние сущности), оказывающие влияние на функционирование системы. Количество потоков на контекстной диаграмме должно быть по возможности небольшим, поскольку каждый из них может быть в дальнейшем разбит на несколько потоков на следующих уровнях диаграммы.

Для сложных систем (признаками сложности могут быть наличие большого количества внешних сущностей (десять и более), распределенная природа системы или ее многофункциональность) строится иерархия контекстных диаграмм. При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных.

Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем. Для проверки контекстной диаграммы можно составить список событий. Список событий должен состоять из описаний действий внешних сущностей (событий) и соответствующих реакций системы на события. Каждое событие должно соответствовать одному или более потокам данных: входные потоки интерпретируются как воздействия, а выходные потоки – как реакции системы на входные потоки. Каждый процесс на DFD, в свою очередь, может быть детализирован при помощи DFD или (если процесс

элементарный) спецификации. Спецификация процесса должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу. Спецификация является конечной вершиной иерархии DFD. Решение о завершении детализации процесса и использовании спецификации принимается аналитиком исходя из следующих критериев:

– наличия у процесса относительно небольшого количества входных и выходных потоков данных (2-3 потока);

– возможности описания преобразования данных процессов в виде последовательного алгоритма;

– выполнения процессом единственной логической функции преобразования входной информации в выходную;

– возможности описания логики процесса при помощи спецификации небольшого объема (не более 20-30 строк).

Создание модели в стандарте DFD

Создание контекстной диаграммы

Методология DFD может быть использована для создания новой модели и для декомпозиции работы. Создадим новую модель работы "Оформление заказов". Для этого в диалоге создания модели (рис.5) выберем тип модели DFD.

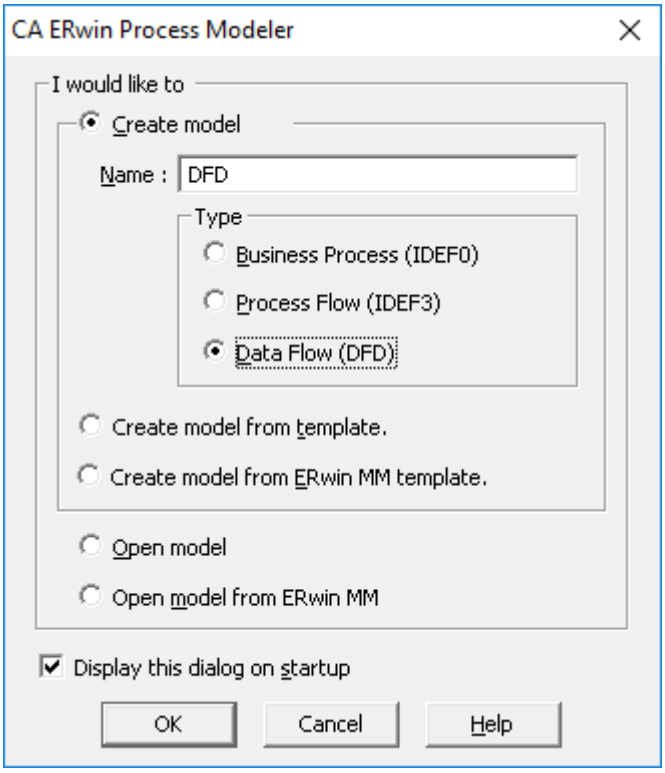


Рис. 5. Выбор типа модели

В открывшемся окне появляется единственная контекстная активность. Обратите внимание, что изображение активности немного отличается от ее изображения в методологии IDEF0: у активности закруглены углы (рис. 6).

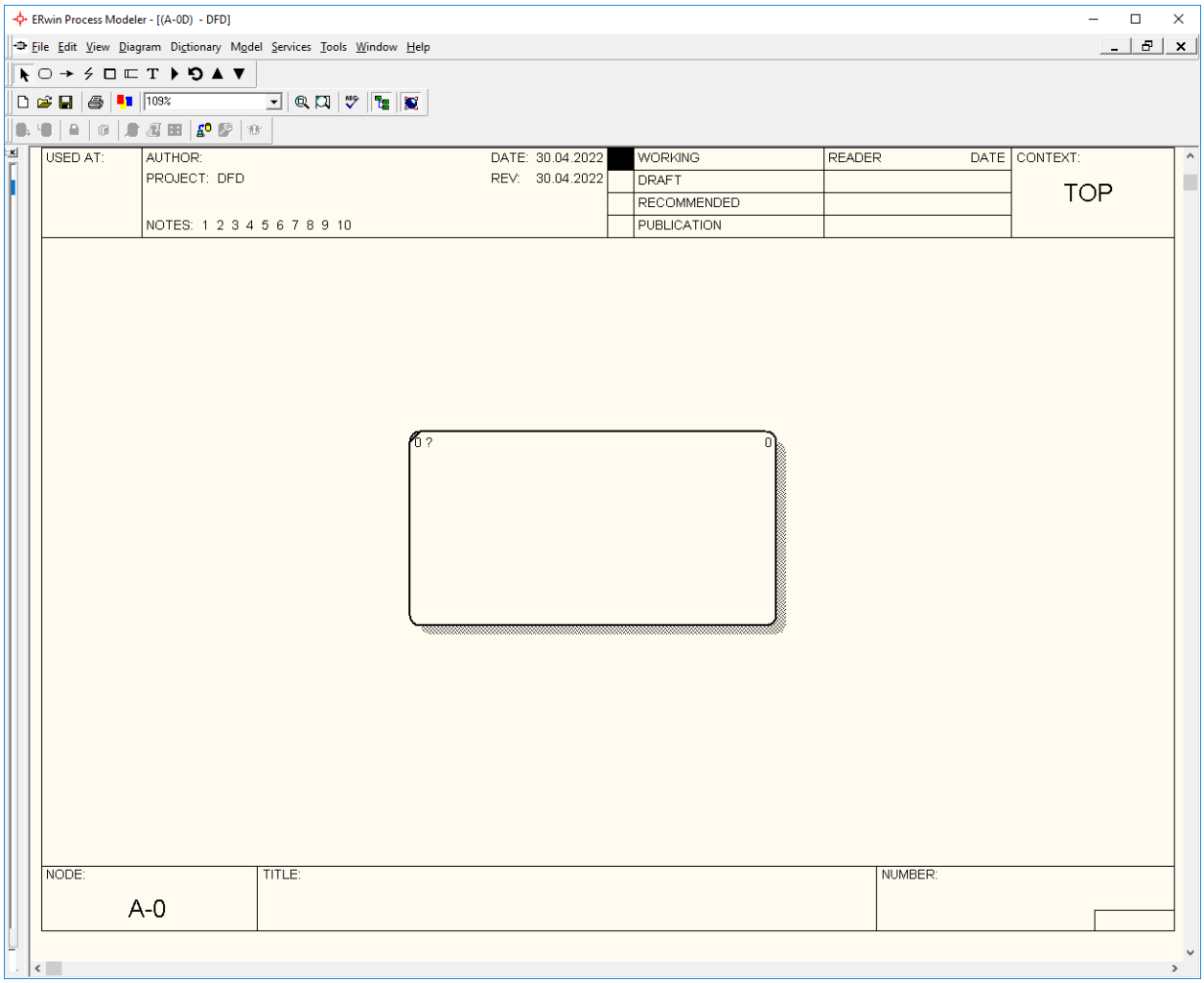


Рис. 6. Контекстная активность

Панель инструментов DFD содержит следующие кнопки:

– кнопка для добавления процесса на диаграмму

– добавление внешней ссылки

– добавление хранилища данных

– проведение новой связи

– инструмент редактирования объектов

– ссылка на пояснение стрелки

– внесение текста в поле диаграммы

– перемещение по моделям с их описанием

– переход между стандартной диаграммой, деревом узлом и FEO

– декомпозиция диаграммы нижнего уровня

– декомпозиция диаграммы верхнего уровня

Построим контекстную диаграмму, как показано на рисунке 7. Зададим имя и свойства активности. Внесем две внешние сущности: источник и приемник. В нашем случае источником и приемником будет одна внешняя сущность "Клиенты". С целью повышения наглядности покажем на диаграмме две внешние сущности с одинаковыми именами. Обратите внимание, что внешние сущности не участвуют в рассматриваемой работе и не подвергаются декомпозиции. Создается внешняя сущность с помощью кнопки.

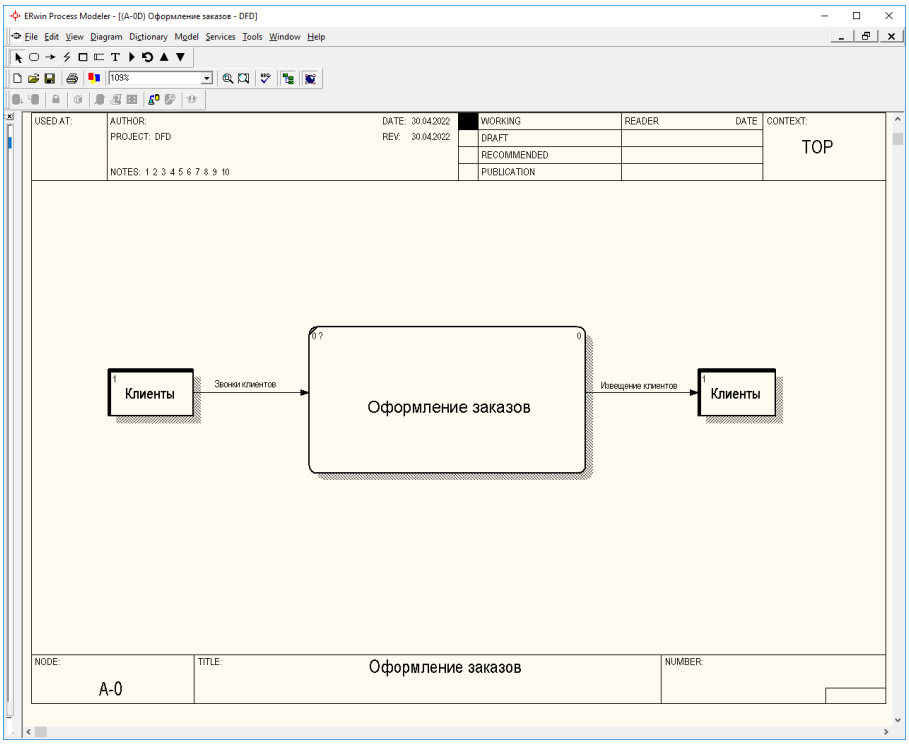


Рис. 7. Контекстная диаграмма

Создание диаграммы декомпозиции

Произведем декомпозицию контекстной диаграммы. Оформление заказа начинается с телефонного звонка клиента. При оформлении заказа необходимо проверить, существует ли клиент в базе данных. Если клиента нет в базе, то необходимо занести данные о клиенте в базу клиентов. Далее производится оформление и внесение заказа в список заказов. При оформлении заказа используются как база клиентов, так и список продуктов. Заканчивается оформление заказа извещением по телефону клиента о результатах оформления заказа. Таким образом, в простейшем случае декомпозиция будет включать две активности: "Проверка и внесение клиента" и "Внесение заказа".

При создании диаграммы декомпозиции в диалоге Activity Box Count следует выбрать тип диаграммы декомпозиции.

Диалог Activity Box Count для методологии DFD (рис. 8)

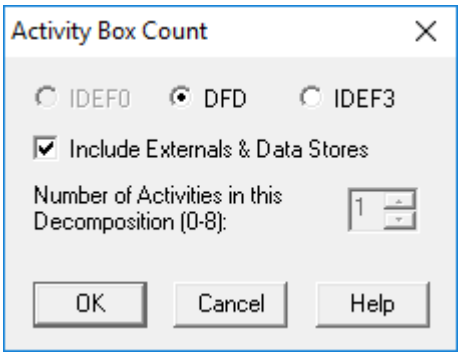


Рис. 8. Диалог Activity Box Count

Свойство Include Externals & Data Stores означает, что на дочернюю диаграмму будут мигрировать внешние сущности и хранилища данных с родительской диаграммы. При этом родительская диаграмма копируется на дочернюю. На дочерней диаграмме надо удалить родительскую активность и создать необходимое число активностей. На родительской диаграмме будут туннелированные стрелки. Можно задать число сущностей на дочерней диаграмме при создании диаграммы декомпозиции. Но в нашем примере выберем миграцию внешних сущностей и хранилищ.

Построим диаграмму декомпозиции, как показано на рисунке 9.

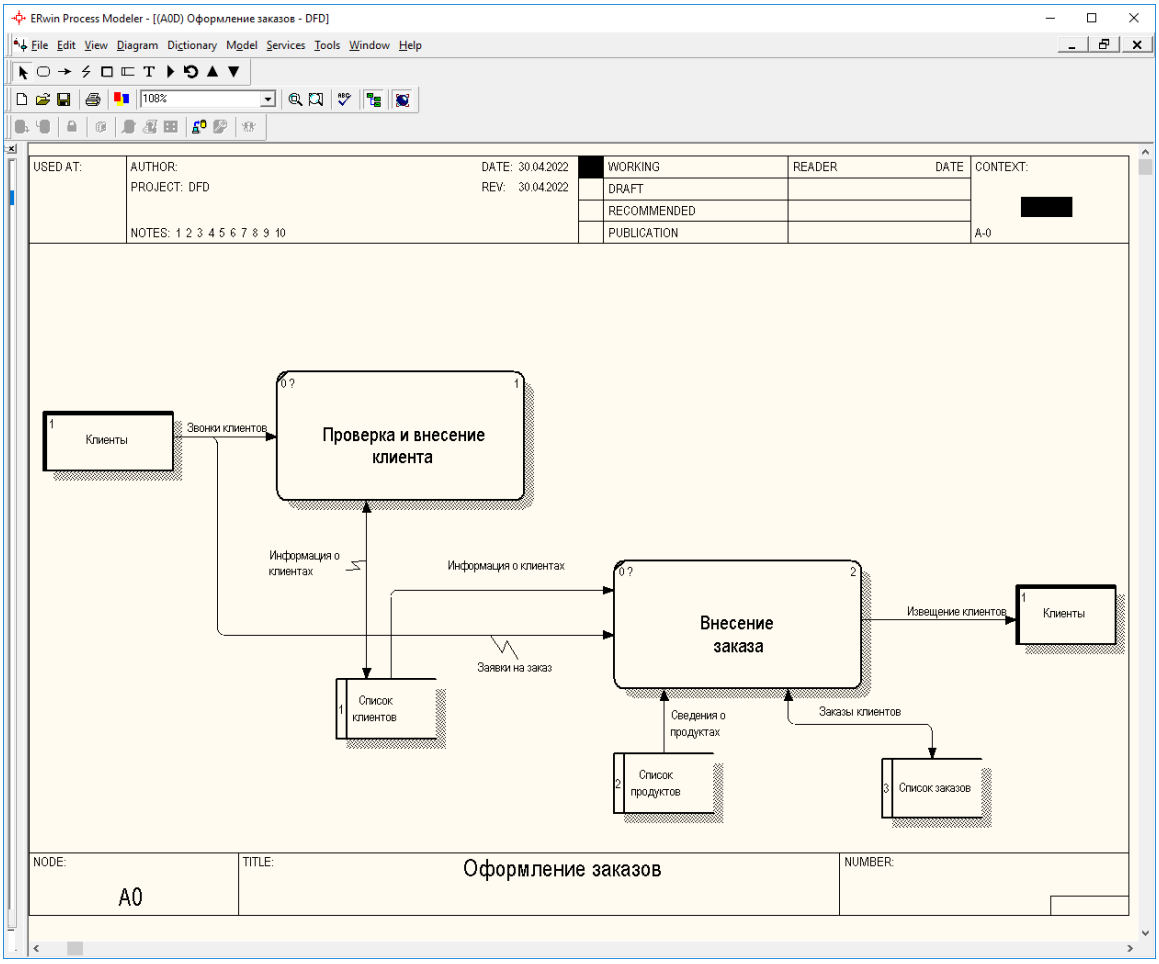


Рис. 9. Диаграмма декомпозиции

Некоторые стрелки на диаграмме декомпозиции являются двунаправленными. Сначала рисуется однонаправленная стрелка. Чтобы сделать стрелку двунаправленной, щелкните правой кнопкой мыши по стрелке, из контекстного меню выберите пункт Style и на вкладке Style меню свойств стрелки выберите вариант двунаправленной стрелки (Bidirectional) (рис. 10).

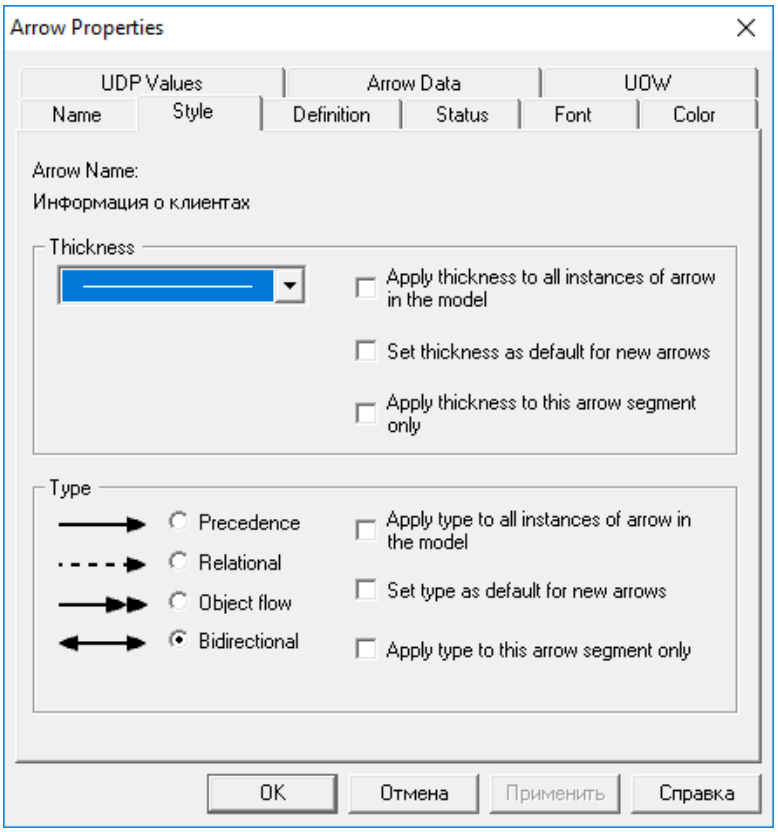


Рис. 10. Создание двунаправленной стрелки

Контрольные вопросы:

1. Для чего служит DFD - диаграмма?
2. В чем отличие DFD - диаграммы от IDFE0?
3. Что содержит в себе диаграмма потоков данных?
4. Дайте объяснение понятий: процесс, внешняя сущность, поток данных, хранилище данных.
5. При каких условиях завершается детализация процесса?

Лабораторная работа №3

Тема: Методология моделирования потоков работ (IDEF3)

Цель: Научиться строить функциональную модель в нотации IDEF3.

Задание: Построить модель предметной области в нотации IDEF3.

Теоретическая часть:

IDEF3 – графический язык описания функциональных систем, позволяющий описать последовательность выполнения работ, направленных на преобразование или обработку какого-либо объекта, а также их временную взаимосвязь и взаимозависимость. IDEF3 является технологией, хорошо приспособленной для сбора данных, требующихся для проведения структурного анализа системы.

В отличие от большинства технологий моделирования бизнес-процессов, IDEF3 не имеет жестких синтаксических или семантических ограничений, делающих неудобным описание неполных или нецелостных систем.

IDEF3 также может быть использован как метод проектирования бизнес-процессов. IDEF3-моделирование органично дополняет традиционное моделирование с использованием стандарта методологии IDEF0.

Если модель в нотации IDEF0 позволяет получить общее представление о функциях, выполняемых моделируемой системой, связях между функциями, механизмах исполнения, то модель в нотации IDEF3 позволяет проследить логику взаимодействия функций, их последовательность и взаимозависимость.

Можно сначала построить функциональную модель в нотации IDEF0, проведя исследования предметной области. Затем, используя полученные знания о предметной области, построить отдельную модель в нотации IDEF3. А можно создать смешанную модель, дополняя по мере необходимости функциональную модель в нотации IDEF0 диаграммами в нотации IDEF3. Также можно дополнять модель DFD диаграммами в нотации IDEF3.

В отличие от IDEF0 нотация IDEF3 не ограничивает автора модели (аналитика) чрезмерно жесткими рамками синтаксиса и семантики, что удобно для описания неполных или не целостных систем, особенно если аналитик плохо знает предметную область.

Основой модели IDEF3 служит так называемый сценарий бизнес-процесса, который выделяет последовательность действий или подпроцессов анализируемой системы. Поскольку сценарий определяет назначение и границы модели, важным является подбор подходящего наименования для обозначения действий. Для подбора необходимого имени применяются

стандартные рекомендации по использованию глаголов и отглагольных существительных, например, «обработать заказ клиента» или «применить новый дизайн».

Сценарий для большинства моделей должен быть документирован. Обычно это название набора должностных обязанностей человека, являющегося источником информации о моделируемом процессе.

Также важным является понимание цели моделирования — набора вопросов, ответами на которые будет служить модель, границ моделирования — какие части системы войдут, а какие не будут отображены в модели, и целевой аудитории — для кого разрабатывается модель.

Главной организационной единицей модели IDEF3 является диаграмма. Диаграммы IDEF3 отображают действие в виде прямоугольника. Действия именуются с использованием глаголов или отглагольных существительных, каждому из действий присваивается уникальный идентификационный номер. Этот номер не используется вновь даже в том случае, если в процессе построения модели действие удаляется. В диаграммах IDEF3 номер действия обычно предваряется номером его родителя (рис.1).

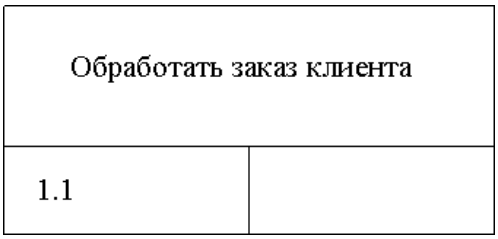


Рисунок 1 - Изображение и нумерация действия в диаграмме IDEF3

Связи выделяют существенные взаимоотношения между действиями. Все связи в IDEF3 являются однонаправленными, и, хотя стрелка может начинаться или заканчиваться на любой стороне блока, обозначающего действие, диаграммы IDEF3 обычно организуются слева направо таким образом, что стрелки начинаются на правой и заканчиваются на левой стороне блоков.

В IDEF3 различают три типа связей –предшествование («Precedence»), отношение («Relation Link»), объектный поток («Object Flow») (таб.1)

Таблица 1 – Типы связей

Изображение	**Название**	**Назначение**
	Предшествование (Precedence)	Исходное действие должно завершиться, прежде чем конечное действие сможет начаться

	Отношение (Relation Link)	Вид взаимодействия между исходным и конечным действиями задается аналитиком отдельно для каждого случая использования такого отношения
	Объектный поток (Object flow)	Выход исходного действия является входом конечного действия. Из этого, в частности, следует, что исходное действие должно завершиться, прежде чем конечное действие сможет начаться

Связь типа *«предшествование»*. Связи этого типа показывают, что исходное действие должно полностью завершиться, прежде чем начнется выполнение конечного действия. **Связь** должна быть поименована таким образом, чтобы человеку, просматривающему модель, была понятна причина ее появления (рис.2).

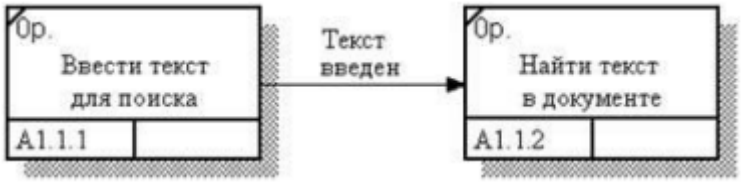


Рисунок 2 – Связь типа «предшествование»

Связь типа *«объектный поток»*. Одна из наиболее часто встречающихся причин использования связи типа «объектный поток» заключается в том, что некоторый объект, являющийся результатом выполнения исходного действия, необходим для выполнения конечного действия. Объект создаётся в некоторой работе и затем неоднократно – в двух или более единицах работа – используется. Обозначение такой связи отличается от связи временного предшествования двойной стрелкой (рис.3).

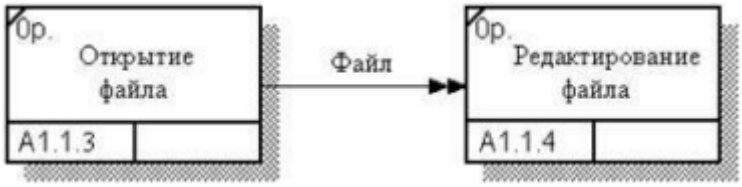


Рисунок 3 – Связь типа «объектный поток»

Связь типа *«отношение»*. Связи этого типа используются для выделения отношений между действиями, которые невозможно описать с использованием предшественных или объектных связей. Значение каждой

такой связи должно быть определено, поскольку связи типа «отношение» сами по себе не предполагают никаких ограничений. Одно из применений отношений — отображение взаимоотношений между параллельно выполняющимися действиями. Обозначается в виде стрелки с пунктирной линией (рис.4).

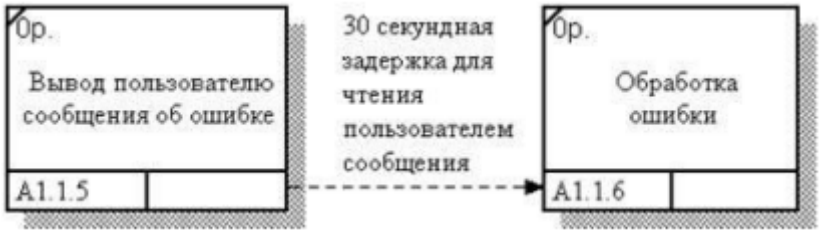


Рисунок 4 – Связь типа «отношение»

Перекрёстки («Junction») – это особые элементы свойственные нотации IDEF3, которые позволяют указать взаимозависимость нескольких работ; различают перекрёстки слияния («Fan-in Junction») и перекрёстки разветвления («Fan-out Junction»); перекрёсток не может использоваться одновременно слияния и разветвления; каждый из перекрёстков получает индивидуальный номер (таб.2).

Таблица 2 - Типы перекрёстков

Обозначение на диаграмме	Наименование	Перекрёсток слияния	Перекрёсток разветвления
&	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 «ИЛИ»	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
X	«ИСКЛЮЧАЮЩЕЕ ИЛИ»	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Перекрёстки слияния показывают, что необходимо завершение всех предыдущих процессов для того, чтобы начался последующий процесс.

Перекрёстки разветвления показывают, что завершение одного процесса ведёт к запуску нескольких последующих процессов.

Различают **асинхронные** и **синхронные перекрёстки**; синхронные – показывают, что все предыдущие или последующие работы должны происходить одновременно, асинхронные допускают неодновременное завершение или начало связанных работ.

По типу выполняемых операций различают перекрёстки типа «И» («AND»), «ИЛИ» («OR») и «ИСКЛЮЧАЮЩЕЕ ИЛИ» («XOR»).

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J (рис.5).

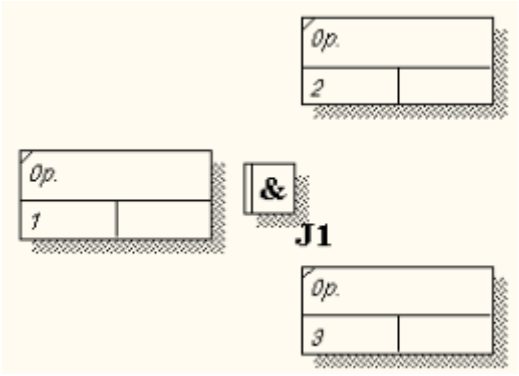


Рисунок 5 – Обозначение нумерации перекрестка

Можно редактировать свойства перекрестка при помощи диалога JunctionProperties, который вызывается из контекстного меню.

Стандартом IDEF3 предусматривается два типа диаграмм – диаграммы описания последовательности этапов процесса («Process Flow Description Diagram» или «PFDD») и диаграммы состояния объекта и трансформации в процессе («Object State Transition Network» или «OSTN»).

На рисунке 4 изображена диаграмма PFDD, являющаяся графическим отображение сценария обработки детали. Прямоугольники на диаграмме PFDD называются функциональными элементами или элементами поведения (Unit of Behavior, UOB) и обозначают событие, стадию процесса или принятие решения. Каждый UOB имеет свое имя, отображаемое в глагольном наклонении и уникальный номер. Стрелки или линии являются отображением перемещения детали между UOB-блоками в ходе процесса (рис.6).

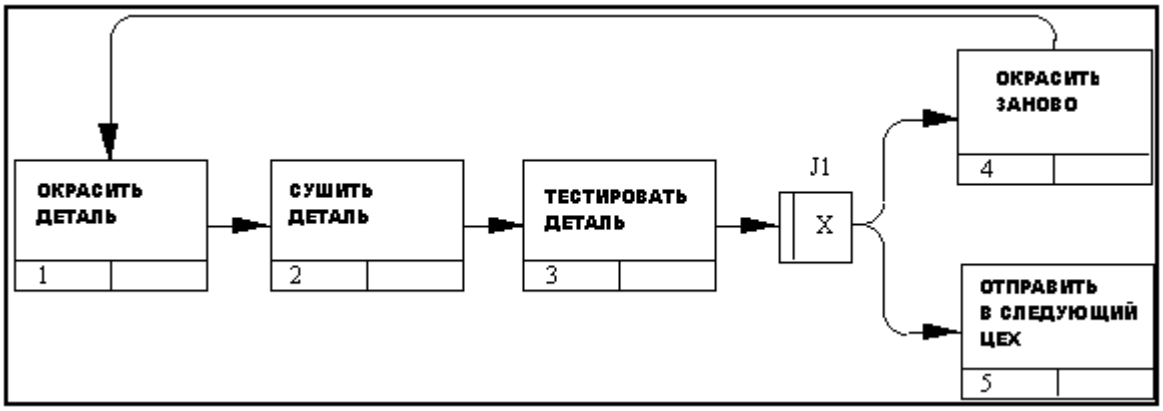


Рисунок 6 - PFDD-диаграмма IDEF3.

Если диаграммы PFDD технологический процесс "С точки зрения наблюдателя", то другой класс диаграмм IDEF3 OSTN позволяет рассматривать тот же самый процесс "С точки зрения объекта". На рисунке 5 представлено отображение процесса окраски с точки зрения OSTN диаграммы. Состояния объекта (в нашем случае детали) и Изменение состояния являются ключевыми понятиями OSTN диаграммы. Состояния объекта отображаются окружностями, а их изменения направленными линиями. Каждая линия имеет ссылку на соответствующий функциональный блок UOB, в результате которого произошло отображаемое ей изменение состояния объекта (рис.7).

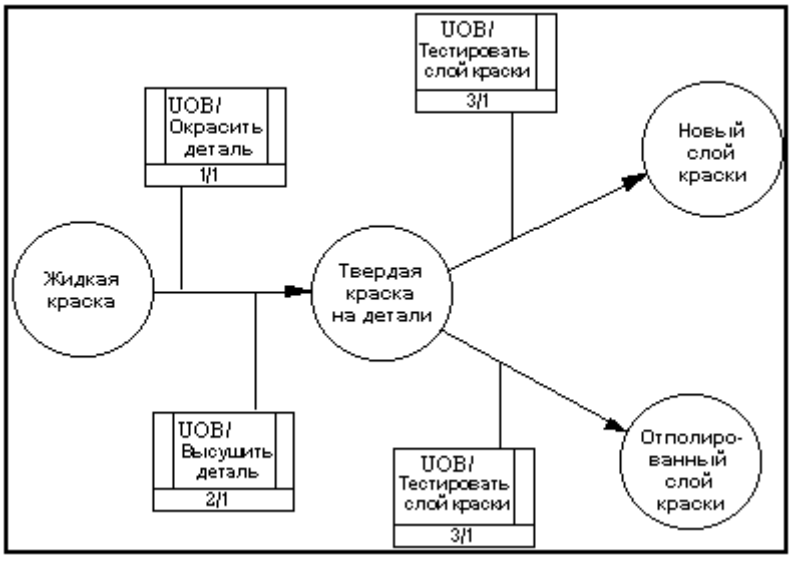


Рисунок 7 - OSTN-диаграмма IDEF3.

Среда BPWin использует только PFDD-нотацию.

Алгоритм создания модели в стандарте IDEF3

Создадим модель «Подготовка специалистов».

Для этого в диалоге создания модели (рис.8) выберем тип модели DFD.

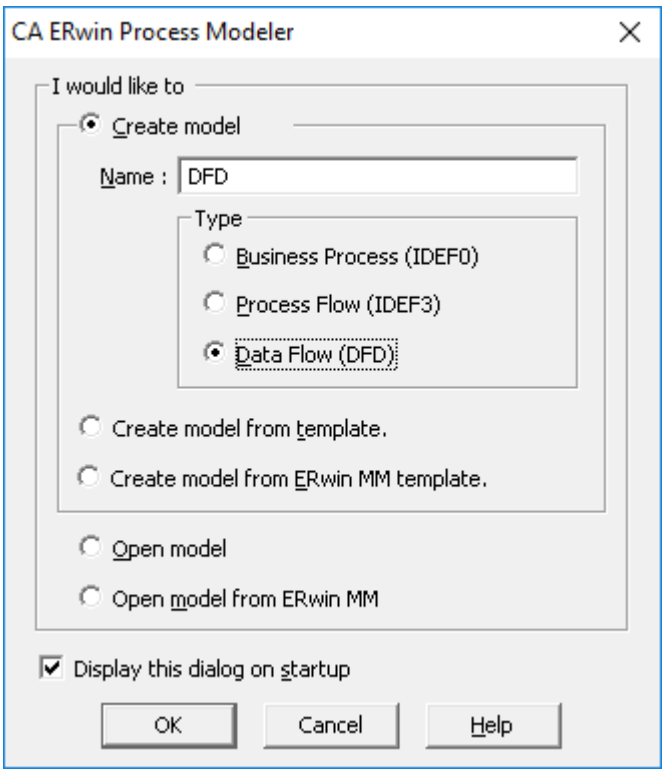


Рис. 8. Выбор типа модели

В открывшемся окне появляется единственная контекстная активность. Обратите внимание, что изображение активности немного отличается от ее изображения в методологии IDEF0: у активности закруглены углы (рис. 9).

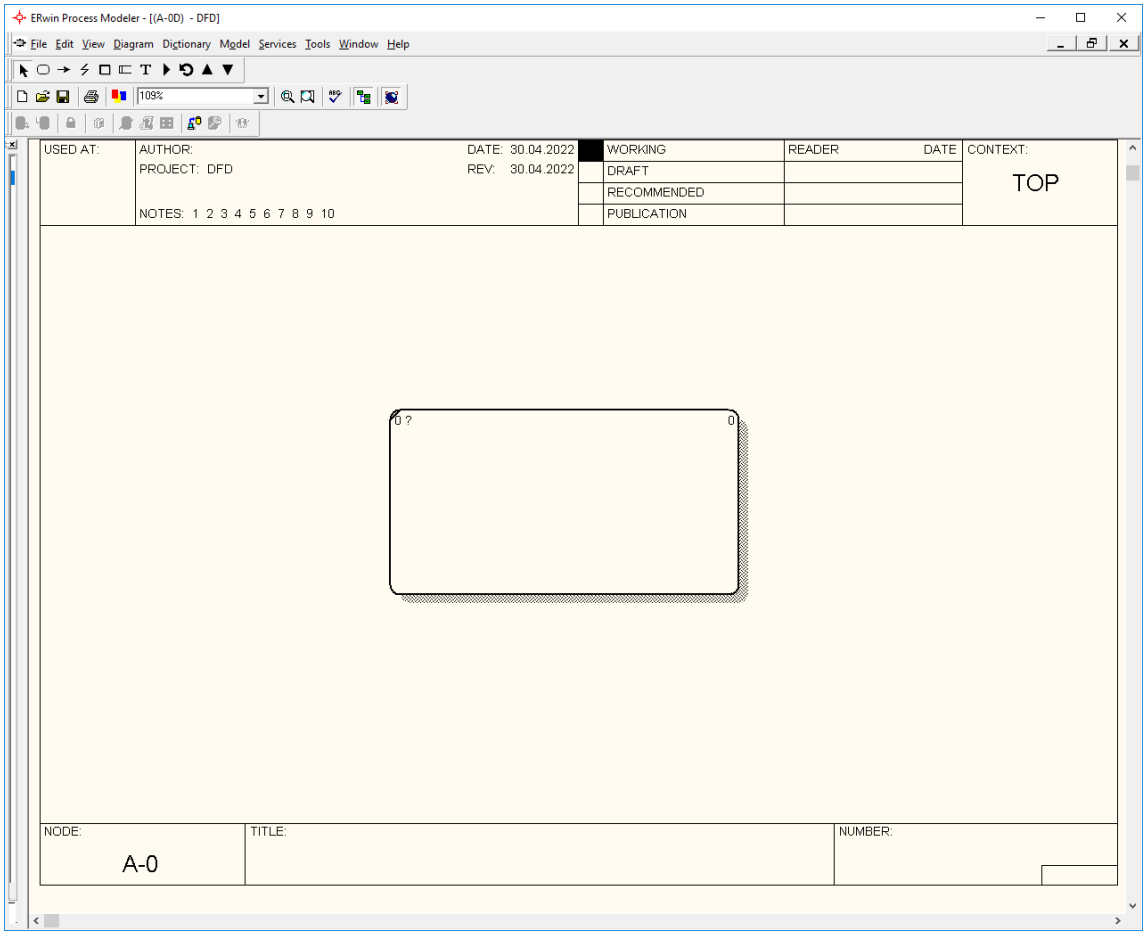


Рис. 9. Контекстная активность

Панель инструментов IDEF3 содержит следующие кнопки:

– кнопка для добавления работы на диаграмму

– проведение новой связи

– внесение перекрестка

– внесение объекта ссылки

– инструмент редактирования объектов

– ссылка на пояснение стрелки

– внесение текста в поле диаграммы

– перемещение по моделям с их описанием

– переход между стандартной диаграммой, деревом узлом и FEO

На контекстной диаграмме щелкните 2 раза мышью по работе. Появится диалог Activity Properties, где во вкладке Name напишите ее имя – Подготовить специалистов.

Во вкладке Font необходимо выбрать Script – кириллический и указать шрифт (рис.10). Галочка в группе Global позволит изменить шрифт для всех объектов модели. Нажмите Применить – ОК.

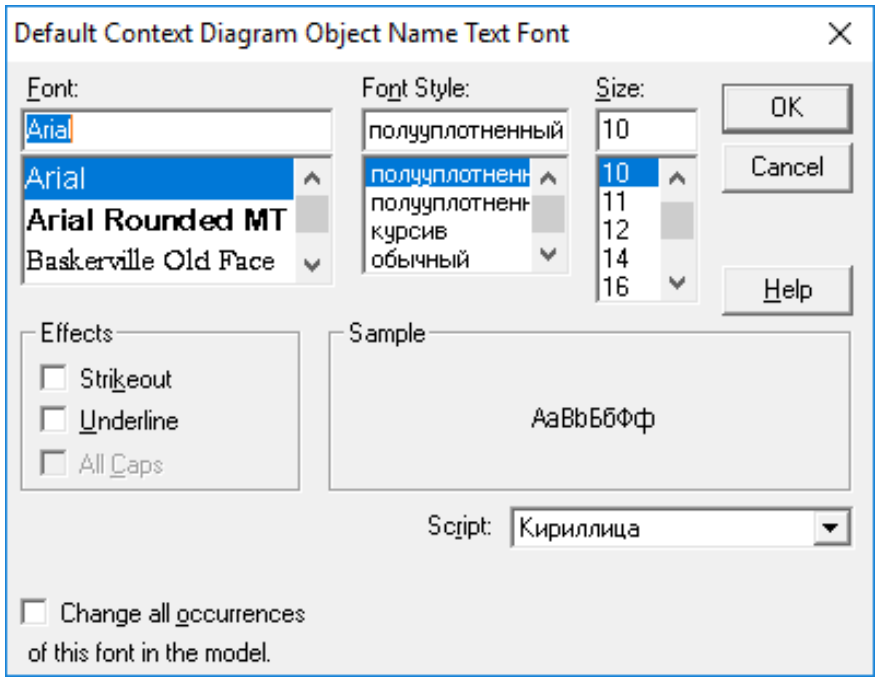


Рис.10. Вкладка Font диалога Activity Properties

На контекстной диаграмме изобразите граничные связи. По сравнению с методологиями IDEF0 и DFD методология IDEF3 не требует обозначения имен связей. В результате получится контекстная диаграмма, показанная на рисунке 11.

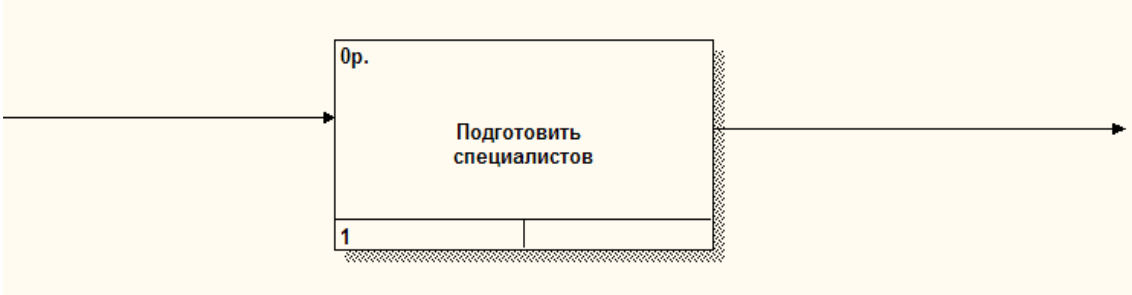


Рис.11. Контекстная диаграмма бизнес-процесса «Подготовить специалистов»

Щелкните один раз по работе готовой контекстной диаграммы. Она выделится черным цветом со своими стрелками.

На панели инструментов IDEF0 нажмите кнопку декомпозиции диаграммы нижнего уровня .

В появившемся диалоге Activity Box Count (рис.12) выберите методологию IDEF3 и количество блоков на диаграмме нижнего уровня – 3.

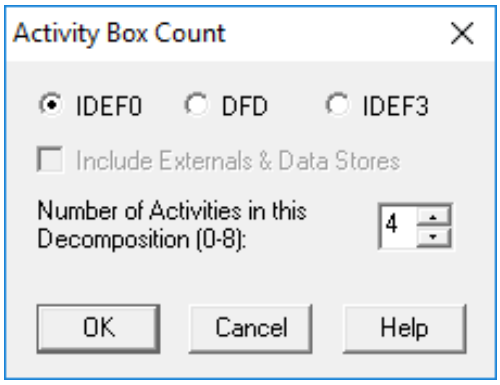


Рис.12. Диалог Activity Box Count

Автоматически будет создана диаграмма декомпозиции. Правой кнопкой мыши щелкните по первой работе, выберите Name и внесите имя работы «Принять абитуриентов». Повторите эти действия с остальными двумя работами, а также свяжите их связями согласно рисунку 13.

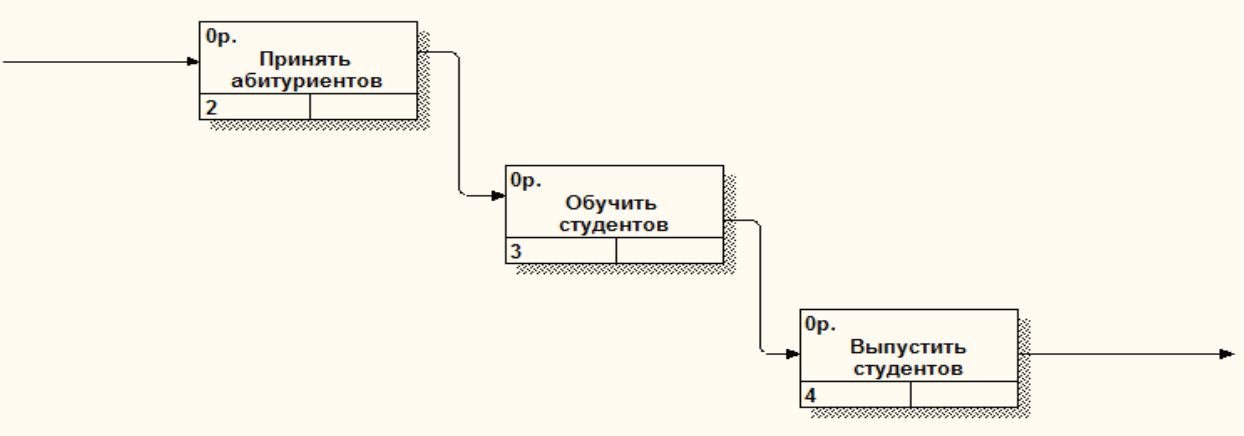


Рис.13. Диаграмма декомпозиции работы «Подготовить специалистов»

Выполните декомпозицию этих трех работ диаграммы согласно рис.14 – 16 соответственно.

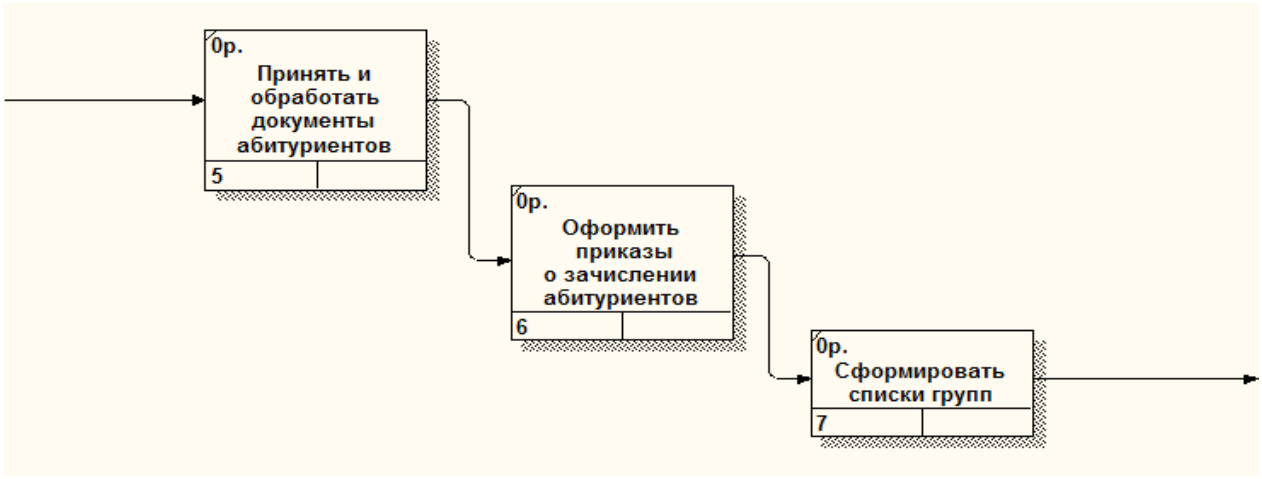


Рис.14. Диаграмма декомпозиции работы «Принять абитуриентов»

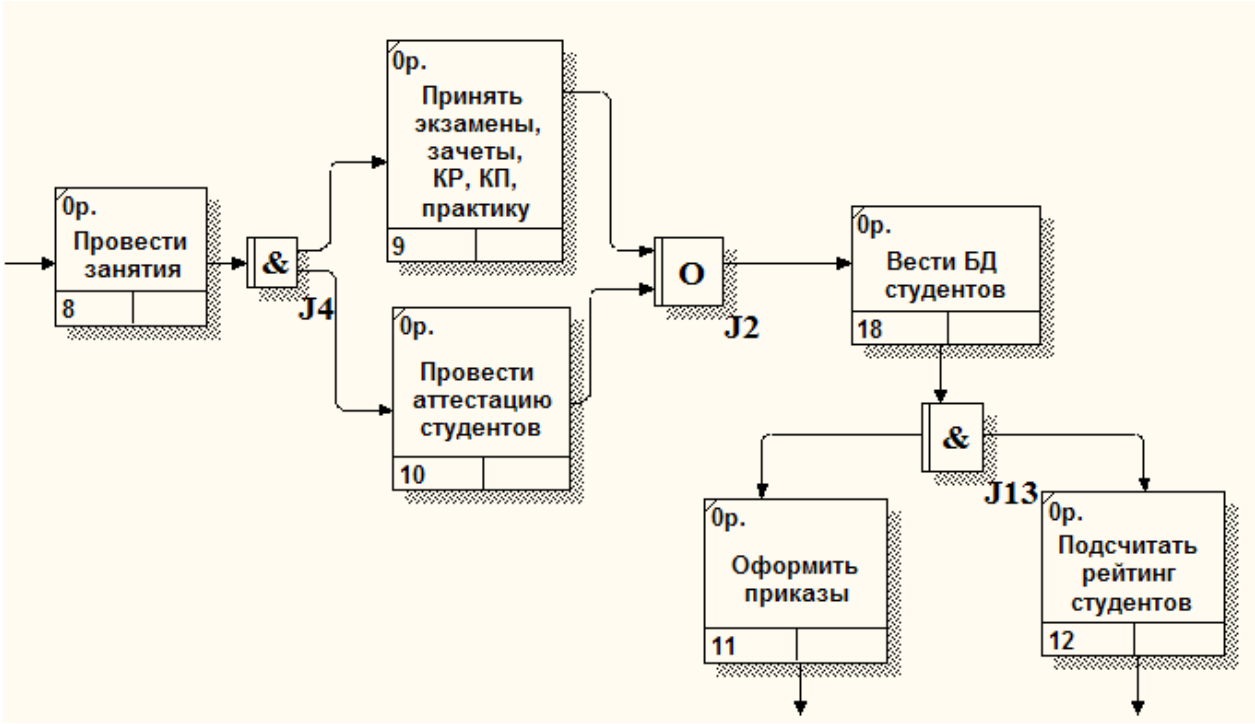


Рис.15. Диаграмма декомпозиции работы «Обучить студентов»

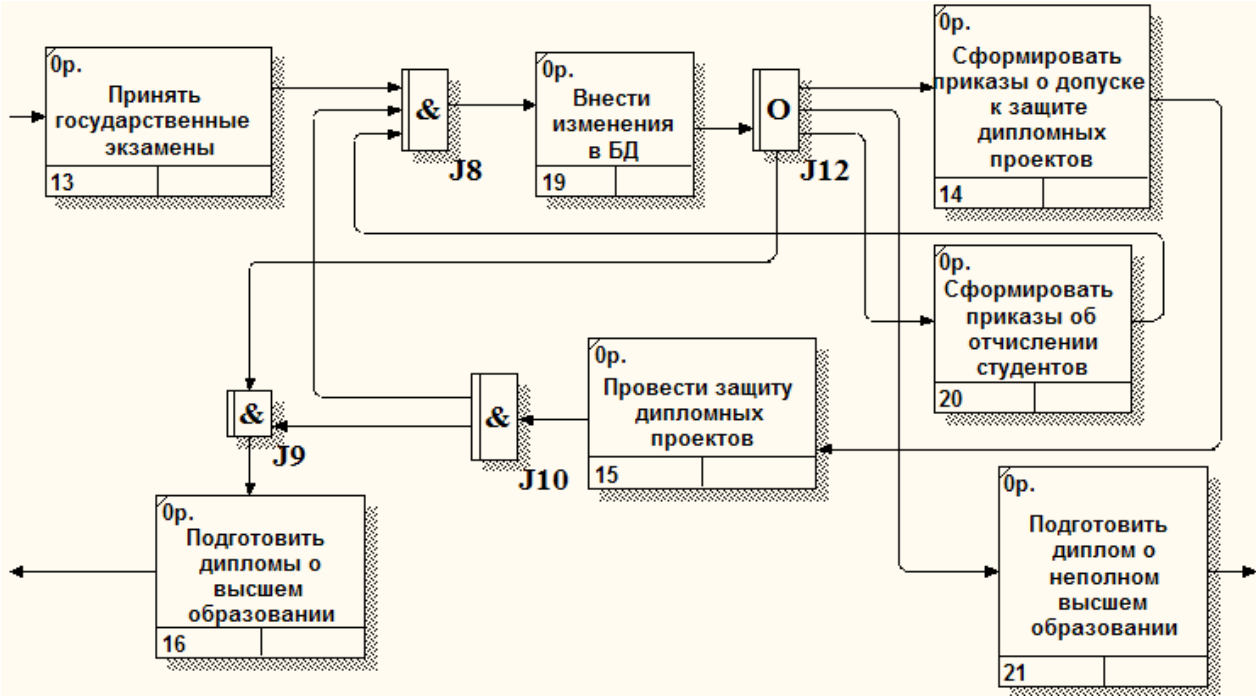


Рис.16. Диаграмма декомпозиции работы «Выпустить студентов»

В результате получилась модель с декомпозицией до 2 уровня.

Контрольные вопросы:

1. Что такое IDEF3? Назначение методологии IDEF3.
2. В чём отличие IDEF3 от IDEF0?
3. Какие виды связей бывают?
4. Дайте определение понятию «перекрестки»
5. Какие виды перекрестков бывают?
6. Какие типы диаграмм существуют в IDEF3?

Лабораторная работа №4

Тема: Диаграмма прецедентов

Цель: знакомство с созданием функциональной модели использования, получение навыков построения диаграмм прецедентов.

Задание:

1. Изучить теоретический материал
2. Построить диаграмму прецедентов по своей индивидуальной теме, выполнив следующие действия:
   - создать диаграмму прецедентов, задав на ней варианты использования и актеров (не менее 2х);
   - добавить отношения между актерами и вариантами использования;
   - добавить описания к актерам и вариантам использования.

Теоретическая часть:

Введение в UML

Для создания моделей анализа и проектирования систем используют языки визуального моделирования, самым популярным из которых на сегодняшний день является UML.

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие выразительных возможностей, этот язык прост для понимания и использования.

Язык UML предназначен прежде всего для разработки программных систем. Его использование особенно эффективно в следующих областях:

- информационные системы масштаба предприятия;
- банковские и финансовые услуги;
- телекоммуникации;
- транспорт;
- оборонная промышленность, авиация и космонавтика;
- розничная торговля;
- медицинская электроника;
- наука;
- распределенные Web-системы.

Разработка модели любой системы всегда предшествует ее созданию или обновлению. Это необходимо хотя бы для того, чтобы яснее представить себе решаемую задачу. Продуманные модели очень важны и для взаимодействия внутри команды разработчиков, и для взаимопонимания с заказчиком. В конце концов, это позволяет убедиться в "архитектурной согласованности" проекта до того, как он будет реализован в коде.

В UML используется четыре вида элементов:

1. фигуры,
2. линии,
3. значки,
4. надписи.

Фигуры используются "плоские" - прямоугольники, эллипсы, ромбы и т. д. Внутри любой фигуры могут помещаться другие элементы нотации. Линии своими концами должны соединяться с фигурами. На UML диаграммах нельзя встретить линий, нарисованных "сами по себе" и не соединяющих фигуры. Применяется два типа линий – сплошная и пунктирная. Линии могут пересекаться, и, хотя таких случаев следует по возможности избегать, в этом нет ничего страшного.

К наиболее известным программа для рисования диаграмм можно отнести:

- IBM Rational Rose;
- Borland Together;
- Gentleware Poseidon;
- Microsoft Visio;
- StarUML.

UML определяет двенадцать типов диаграмм, разделенных на три группы:

- четыре типа диаграмм представляют статическую структуру приложения;
- пять представляют поведенческие аспекты системы;
- три представляют физические аспекты функционирования системы.

Базовые отношения или связи в языке UML

——	Отношение ассоциации (association relationship)	Отношение ассоциации показывает, что один класс каким-то образом связан с другим классом. Изображается сплошной линией, соединяющей классы.
---->	Отношение зависимости (dependency relationship)	Отношение зависимости показывает, что изменение одного класса влечет изменение другого класса. Чаще всего применяется, когда один класс использует другой в качестве аргумента. Изображается пунктирной линией со стрелкой, направленной от зависимого класса к независимому.
—▷	Отношение обобщения (generalization relationship)	Это отношение между общей сущностью (суперклассом, или родителем) и ее конкретным воплощением (субклассом, или потомком). Обобщения иногда называют отношениями типа "является", имея в виду, что одна сущность является частным выражением другой, более общей. Обобщение означает, что объекты класса-потомка могут использоваться всюду, где встречаются

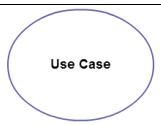
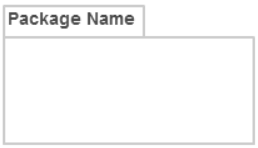
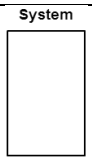
		объекты класса-родителя, но не наоборот. Изображается в виде линии с большой незакрашенной стрелкой.
	Отношение реализации (realization relationship)	Это отношение между двумя элементами модели, при котором один элемент (клиент) реализует поведение, заданное другим (поставщиком). Изображается в виде пунктирной линии с большой незакрашенной стрелкой, указывающей на поставщика. Чаще всего реализации используют для определения отношений между интерфейсом и классом или компонентом, который предоставляет объявленные в интерфейсе операции или услуги.
	Отношение агрегации	*Отношение агрегации* - частный случай ассоциации. Представляет собой отношение типа «целое/часть». Изображается в виде простой ассоциации с незакрашенным ромбом со стороны «целого». Агрегация не является наследованием, поскольку все классы-«части» в агрегации являются вполне самостоятельными, со своими атрибутами и операциями, отличающимися от атрибутов и операций класса-«целое».
	Отношение композиции	*Отношение композиции* - частный случай агрегации. Служит для выделения специальной формы отношения «целое-часть», при которой составляющие части в некотором смысле находятся внутри целого. Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его составные части. Изображается в виде сплошной линии с закрашенным ромбом возле класса «целое».

Диаграмма прецедентов (по-другому «Диаграмма вариантов использования») – это тип поведенческой диаграммы UML, который часто используется для анализа различных систем. Они позволяют визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой.

Диаграммы прецедентов используются для сбора требований к использованию системы. В зависимости от ваших требований вы можете использовать эти данные различными способами.

Для построения диаграммы прецедентов необходимы следующие объекты:

Вид объекта	**Название объекта**	**Назначение объекта**

	Use Case (Прецедент)	Добавление на диаграмму нового варианта использования (прецедента) - представляет собой функцию или действие внутри системы.
	Actor (Актер)	Добавление на диаграмму нового актера — это любая сущность, которая выполняет роль (человек, организация или внешняя система) в одной данной системе
	Package (Пакет)	Добавление на диаграмму нового пакета – дополнительным элементом, который чрезвычайно полезен в сложных диаграммах, используются для группировки вариантов использования.
	System (Контейнер)	Добавление на диаграмму нового контейнера (системы). Система используется для определения сферы применения и нарисована в виде прямоугольника. Это необязательный элемент, но полезный при визуализации больших систем.

Построение таких диаграмм желательно осуществлять с помощью CASE-средств или онлайн конструктора – Lucidchart

Пример диаграммы прецедентов работы *Компьютерной фирмы*.

Для нашей предметной области выделяем следующих актеров:

Актер	Краткое описание
Менеджер по работе с клиентами	Сотрудник, который общается с заказчиком и работает с заказом
Менеджер по снабжению	Сотрудник, который занимается закупкой необходимых комплектующих
Инженер по сборке настольных компьютеров	Сотрудник, который занимается сборкой настольных компьютеров
Инженер по сборке ноутбуков	Сотрудник, который занимается сборкой ноутбуков

Инженер по тестированию	Сотрудник, который занимается тестированием собранных компьютеров
Завскладом	Сотрудник, который заведует складом комплектующих

Выделим следующие прецеденты:

Прецедент	**Краткое описание**
Работа с заказом	Запускается менеджером, по работе с клиентами. Позволяет вносить, изменять, удалять или просматривать заказ.
Управление информацией о клиенте	Запускается менеджером по работе с клиентами. Позволяет добавлять, изменять или удалять клиентов, а также просматривать информацию о клиентах.
Управление информацией о поставщиках	Запускается менеджером по снабжению. Позволяет добавлять, изменять или удалять поставщиков, а также просматривать информацию о поставщиках.
Управление информацией о комплектующих	Запускается менеджером по снабжению. Позволяет просматривать информацию о комплектующих, производить анализ их расходования, прогнозировать необходимое их количество и делать заказ.
Сборка компьютеров	Запускается инженером по сборке. Позволяет просматривать наряды на сборку компьютеров и делать отметки о ходе выполнения работы.
Требование необходимых комплектующих	Запускается инженером по сборке. Предназначено для затребования необходимых комплектующие со склада.
Тестирование компьютеров	Запускается инженером по тестированию. Позволяет просмотреть список компьютеров, подлежащих тестированию и сделать отметки о ходе выполнения работ.
Учет поступления и выдачи комплектующих	Запускается завскладом. Позволяет вести учет поступления и выдачи запчастей и комплектующих.

Созданная диаграмма прецедентов показана на рисунке 1.

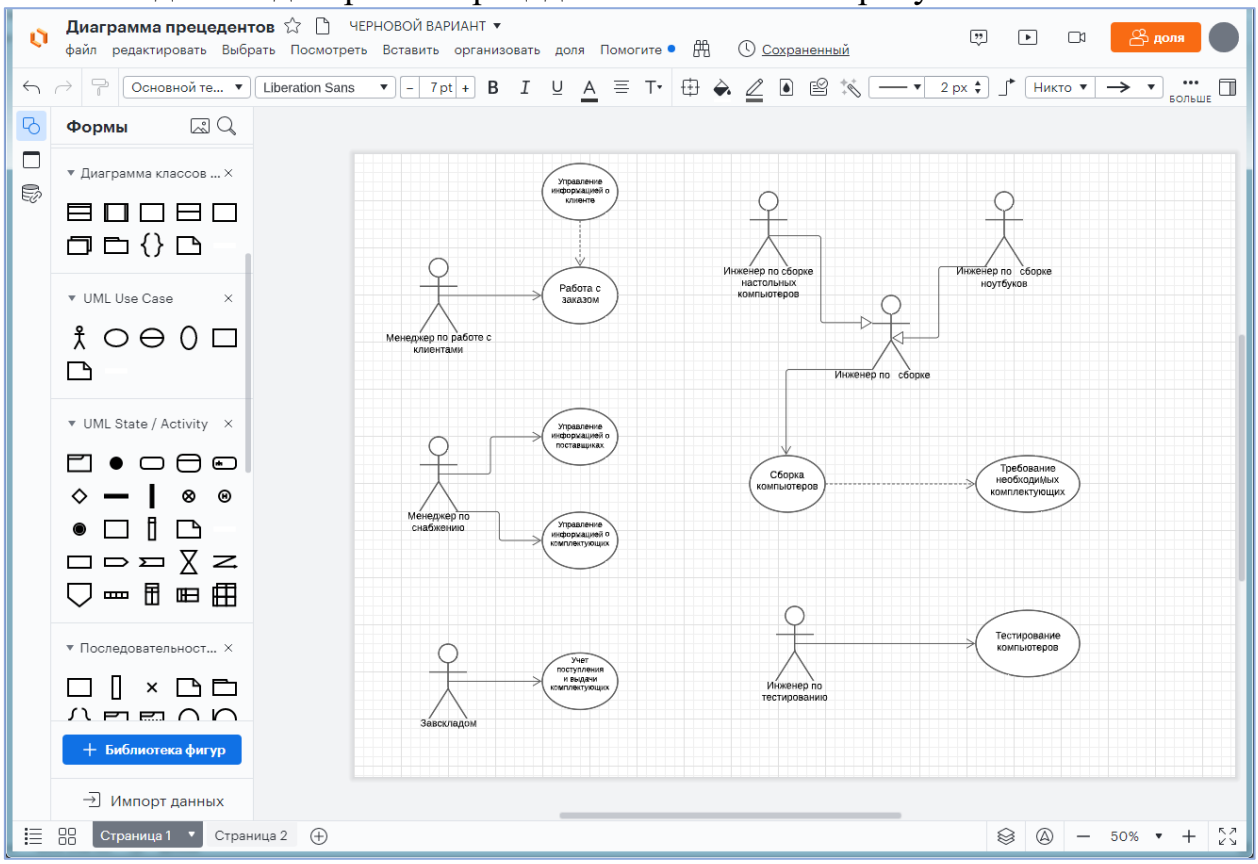


Рис.1. Диаграмма прецедентов

Рассмотрим теперь отношения между актерами и прецедентами:

1. Всех актеров мы связали с прецедентами *отношением ассоциации.*

2. Для прецедента *Сборка компьютеров* не имеет значение какой именно актер будет с ним взаимодействовать - *Инженер по сборке настольных компьютеров* или *Инженер по сборке ноутбуков.* Поэтому мы ввели еще одного актера - *Инженер по сборке*, с которым связали первых двух актеров *отношением обобщения.*

3. Отношение между прецедентами *Работа с заказом* и *Управление информацией о клиенте* - *отношение зависимости*, поскольку, когда актер *Менеджер по работе с клиентами* работает с заказом (оформляет, меняет и т.д.), то не всегда при этом он управляет информацией о клиентах.

4. Отношение между прецедентами *Сборка компьютеров* и *Требование необходимых комплектующих* - *отношение зависимости*, поскольку для сборки компьютеров обязательно нужно заказывать необходимые комплектующие со склада.

Контрольные вопросы.

1. В чем смысл варианта использования?
2. Назначение диаграммы вариантов использования.
3. Назовите основные объекты диаграмм вариантов использования.
4. Что такое действующее лицо (актер)?

5. Какую роль могут играть действующие лица (актеры) по отношению к варианту использования (прецеденту)?

6. Какие типы отношений между актерами и прецедентами существуют? Назначение отношения обобщения.

Лабораторная работа №5

Тема: Диаграмма взаимодействия.

Цель: ознакомление с созданием моделей, описывающих поведение взаимодействующих групп объектов, получение навыков построения диаграмм взаимодействия.

Задание:

3. Изучить теоретический материал
4. Реализовать диаграмму деятельности, выполнив следующие действия:

– создать диаграмму последовательности, описывающую один из бизнес-процессов выбранной предметной области;

Теоретическая часть

Диаграммы взаимодействия являются моделями, описывающими поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой в рамках одного варианта использования.

Существует два вида диаграмм взаимодействия: диаграммы последовательности (sequencediagrams) и кооперативные, или сотрудничества (collaboration diagrams).

Рассмотрим подробно диаграммы последовательности.

Диаграммы последовательности – это диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определенного объекта и взаимодействие актеров ИС в рамках какого-либо определённого прецедента.

Диаграмма последовательностей состоит из обычных объектов, представленных в виде прямоугольников, сообщений, изображенных сплошными линиями со стрелками, а также вертикальной оси времени, определяющей последовательность событий.

Объекты располагаются в верхней части диаграммы слева направо. Под каждым объектом расположена пунктирная вертикальная линия, которая называется линией жизни этого объекта. Вдоль линии жизни располагаются узкие длинные прямоугольники, которые называются точками активации. Точка активации представляет выполнение объектом некоторой операции. Длина прямоугольника соответствует длительности процесса активации.

Сообщения, передаваемые от одного объекта к другому, на диаграмме изображаются в виде линий, соединяющих линии жизни этих объектов. Объект может передать сообщение самому себе. Существует 3 типа сообщений:

– Вызов – запрос объекта-отправителя к объекту-получателю на выполнение одной из его операций. Обычно отправитель ждет завершение выполнения операции. Такой тип сообщений называется синхронным, т.к. в этом случае, отправитель «синхронизирует» свои действия с получателем

– Если сообщение асинхронное, то отправитель передает управление получателю и не ожидает ответа для продолжения выполнения своих действий.

– Ответ – ответ объекта-получателя сообщения Вызов объекту-отправителю.

Время на диаграмме изменяется вдоль вертикального направления. Начало отсчета находится вверху, увеличение времени происходит сверху вниз. Чем позже передается сообщение, тем ближе к нижней части диаграммы последовательностей оно располагается.

Построение таких диаграмм желательно осуществлять с помощью CASE-средств (StarUML, Rational Rose, MS Visio и др.) или онлайн конструкторов – Lucidchart и Creately.

Пример диаграммы последовательности

Создание диаграммы последовательности для процесса бухгалтерского учета и отчетности.

В представленном примере объектами являются запросы предприятие, налоговая инспекция и банк, обозначенные прямоугольниками, а также агенты руководитель и главный бухгалтер, обозначенные элементом «Актер».

Линия жизни идет вертикально вниз от каждого объекта и упорядочивает сообщения таким образом, чтобы они читались сверху вниз. Каждая линия жизни имеет полосу активности (цветные полосы), показывающую интервал активности участника при взаимодействии

Сообщения показывают взаимодействие между объектами в виде горизонтальной стрелки, концы которой лежат на линиях жизни. Направление стрелки указывает на адресата, а положение на линии жизни упорядочивает сообщения по времени.

Результат построения диаграммы показан на рисунке 1.

Рис.1. Диаграмма последовательности «Выбрать и оплатить книгу»

Таким образом, UML-диаграмма последовательности позволяет достаточно наглядно показать взаимодействие между разными объектами, детализируя какими сигналами прямыми и ответными они обмениваются.

Контрольные вопросы

1. Что собой представляют диаграммы взаимодействия?
2. Какие объекты применяются для построения диаграмм взаимодействия?

Лабораторная работа №6

Тема: Диаграмма классов

Цель: изучить принципы работы с классами, получить навыки построения диаграмм классов, создания пакетов и группировки классов в пакеты.

Задание

1. Изучить теоретический материал
2. Реализовать предлагаемую в работе диаграмму классов, выполнив следующие действия:
   - создать диаграмму классов для одного из сценариев диаграммы прецедентов, созданной в предыдущей лабораторной работе
   - для каждого класса задать атрибуты и операции
   - добавить текстовое описание самого класса, описания его атрибутов и операций

Теоретическая часть:

Диаграмма классов - это набор статических, декларативных элементов модели. Диаграммы классов могут применяться и при прямом проектировании, то есть в процессе разработки новой системы, и при обратном проектировании - описании существующих и используемых систем.

Диаграмма класса показывает классы и их отношения, тем самым представляя логический аспект проекта. Каждый класс должен иметь имя. Имя каждого класса должно быть уникально в содержащем его проекте. Диаграмма классов определяет этапы объектов системы и различные статистические связи, которые существует между ними. Имеется два основных вида статистических связей:

– ассоциации (например, менеджер может вести несколько проектов);

– подтипы (работник является разновидностью личности).

На диаграммах классов также изображаются атрибуты классов, операции и ограничения, которые накладываются на связи между объектами.

Класс (class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции (рис.1). В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).

Рис. 1. Графическое изображение класса на диаграмме классов

Имя класса должно быть уникальным в пределах пакета, который описывается некоторой совокупностью диаграмм классов (возможно, одной диаграммой). Оно указывается в первой верхней секции прямоугольника. В дополнение к общему правилу наименования элементов языка UML, имя класса записывается по центру секции имени полужирным шрифтом и должно начинаться с заглавной буквы. Класс может не иметь экземпляров или объектов. В этом случае он называется абстрактным классом, а для обозначения его имени используется наклонный шрифт (курсив). В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом. Данное обстоятельство является семантическим аспектом описания соответствующих элементов языка UML.

Атрибуты

Во второй сверху секции прямоугольника класса записываются его атрибуты (attributes) или свойства. В языке UML принята определенная стандартизация записи атрибутов класса, которая подчиняется некоторым синтаксическим правилам. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения.

Квантор видимости может принимать одно из трех возможных значений и, соответственно, отображается при помощи специальных символов:

– Символ "+" обозначает атрибут с областью видимости типа общедоступный (public). Атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма.

– Символ "#" обозначает атрибут с областью видимости типа защищенный (protected). Атрибут с этой областью видимости недоступен или невиден для всех классов, за исключением подклассов данного класса.

– Знак "-" обозначает атрибут с областью видимости типа закрытый (private). Атрибут с этой областью видимости недоступен или невиден для всех классов без исключения.

Операция

Операция (operation) представляет собой некоторый сервис, предоставляющий каждый экземпляр класса по определенному требованию. Совокупность операций характеризует функциональный аспект поведения класса. При этом каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции имени операции, выражения типа возвращаемого операцией значения и, возможно, строка-свойство данной операции.

Пример диаграммы классов *работы театра*.

В представленной диаграмме классов показаны основные элементы предметной области, а также их атрибуты и операции.

Класс Театр включает в себя следующие атрибуты:

Код театра
Название театра
Вид театра
Директор театра
И операции:
Добавить()
Обновить()
Удалить()

Данный класс необходим для описания общих сведений о театрах, которые предлагают свои билеты на продажу.

Класс Спектакль отражает перечень всех спектаклей во всех театрах и включает атрибуты:

Код спектакля
Название спектакля
Вид актера
Постановщик
И операции этого класса:
Открыть()
Закрыть()
Изменить()

Класс Афиша зависит от класса Спектакль. Атрибуты класса Афиша:

Код спектакля
Код театра
Дата
Операции:
Добавить()
Удалить()
Убрать()

Класс Билеты содержит все билеты на все спектакли и включает следующие атрибуты:

Код билета
Дата
Цена
Операции данного класса:
Заказать
Отменить

Также в базе данных имеются данные о всех сотрудниках театра именно это отражает класс Сотрудники. Атрибуты:

Код сотрудника
Фамилия
Имя
Отчество
Операции:
Принять()
Изменить()

Уволить()
Класс Жанр. Атрибуты:
Код жанра
Название
Описание
Описание:
Добавить()
Удалить()
Обновить()

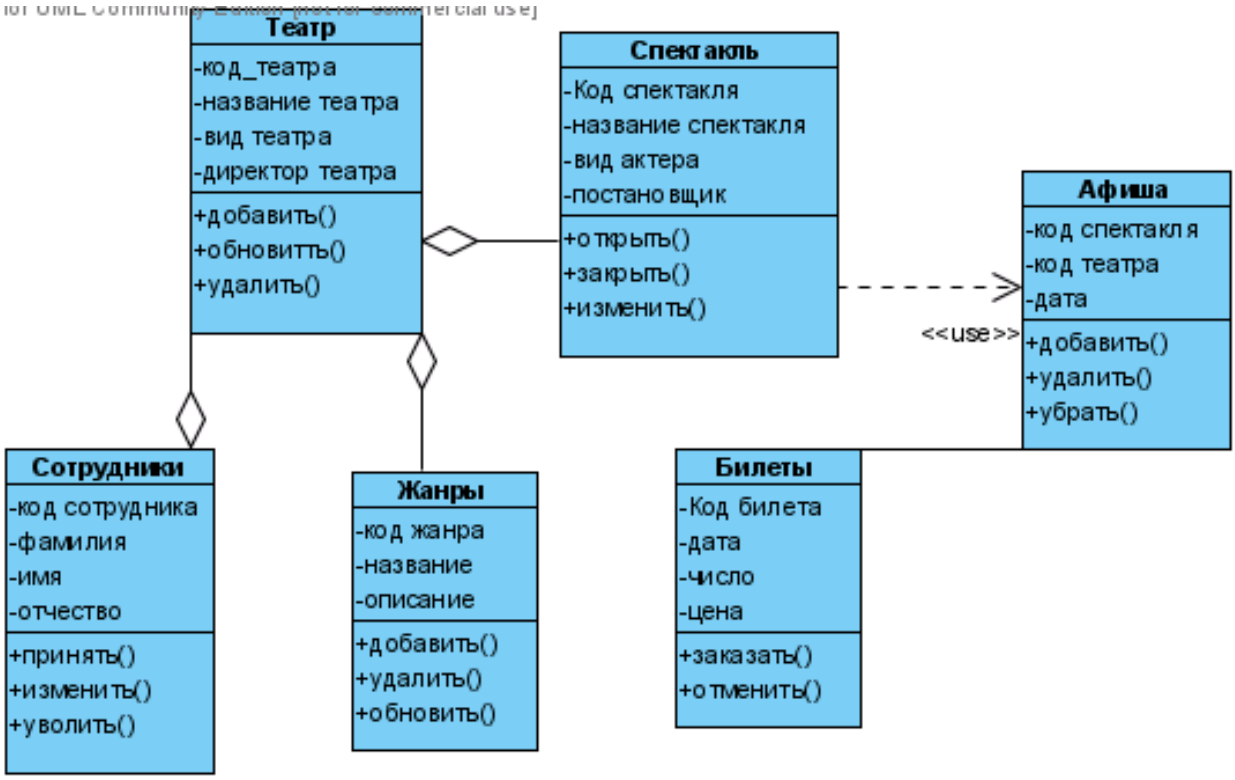


Рис.2. Диаграмма классов работы театра

Контрольные вопросы

1. Назначение диаграммы классов.
2. Назовите основные компоненты диаграммы классов.
3. Что собой представляет ассоциация?
4. Как описывается класс?
5. Что входит в описание атрибута?
6. Что представляет собой операция класса?

Лабораторная работа №7

Тема: Диаграмма деятельности (активности)

Цель: изучить принципы построения диаграмм деятельности (активности); получить навыки построения диаграмм.

Задание:

5. Изучить теоретический материал
6. Реализовать диаграмму деятельности, выполнив следующие действия:

– создать диаграмму деятельности (активности), описывающую один из бизнес-процессов выбранной предметной области.

Теоретическая часть

Диаграммы деятельности можно использовать на всех этапах разработки программного обеспечения и для различных целей. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

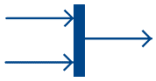
Диаграмма активности позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему.

Диаграммы активности также могут быть использованы для отображения потока событий в бизнес-процессе. Они могут быть использованы для изучения бизнес-процессов с целью определения их потока и требований.

Ниже приведены часто используемые объекты диаграммы деятельности с пояснениями (таб.1).

Таблица 1. Объекты диаграммы деятельности

Объект	Имя	Использовать
	Пуск/ начальный узел	Используется для представления начальное состояние деятельности
	Состояние действия	Используется для представления деятельности процесса
	Действие	Используется для представления исполняемых действий

	Поток управления / Край	Используется для представления потока управления от одного действия к другому
	Конечный узел активности	Используется для обозначения конца всех контрольных потоков в рамках деятельности
	Поток конечный узел	Используется для обозначения конца одного потока управления
	Узел принятия решений	Используется для представления условной точки ответвления с одним входом и несколькими выходами
	Узел слияния	Используется для представления слияния потоков. Он имеет несколько входов, но один выход.
	Вилка (разделение)	Используется для представления потока, который может разветвляться на два и более параллельных потока
	Слияние	Используется для представления двух входов, которые объединяются в один выход
	Отправка сигнала	Используется для представления действия по отправке сигнала на приемную деятельность

	Получение сигнала	Используется для обозначения того, что сигнал получен
	Примечание/комментарий	Используется для добавления соответствующих комментариев к элементам

В Диаграммы деятельности также используются разделы для представления или группирования действий, выполняемых различными действующими лицами в одном потоке. Благодаря этому легко определить, каким из объектов выполняется каждая из активностей. Имя дорожки может означать роль или объект, которому она соответствует (рис.1).

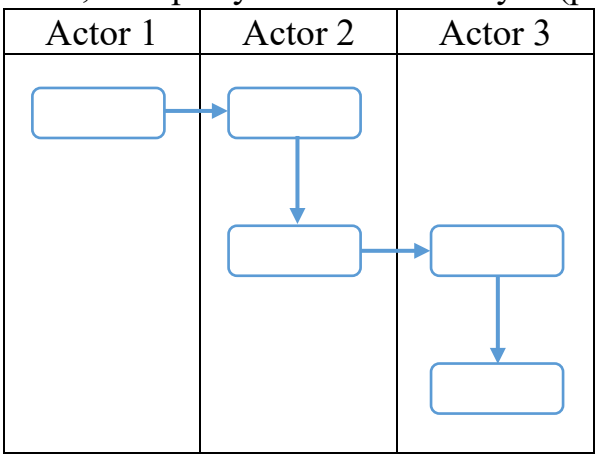


Рис.1. «Дорожки» Actor 1, Actor 2 и Actor 3

Построение таких диаграмм желательно осуществлять с помощью CASE-средств (StarUML, Rational Rose, MS Visio и др.) или онлайн конструктора – Lucidchart

Пример диаграммы деятельности (активности) *Компьютерной фирмы*.

Создание диаграммы деятельности для бизнес-процесса предприятия по сборке компьютеров.

Рассмотрим в целом, что происходит на предприятии от момента оформления заказа на сборку компьютера до выдачи готового компьютера. После оформления заказа менеджер по работе с клиентами передает его менеджеру по сборке, который, прежде чем начать сборку, заказывает необходимые комплектующие со склада. На складе заведующий подбирает необходимые комплектующие (в случае их отсутствия заказывает их у менеджера по снабжению) и передает их инженеру по сборке. После получения комплектующих менеджер по сборке осуществляет сборку

компьютера и передает его инженеру по тестированию. Если компьютер не прошел тестирование, он возвращается для повторной сборки. При успешном завершении тестирования компьютер передается на склад на хранение. Со склада компьютер по требованию передается инженеру по работе с клиентами, который оформляет на него документы и выдает клиенту.

Результат построения диаграммы показан на рисунке 2.

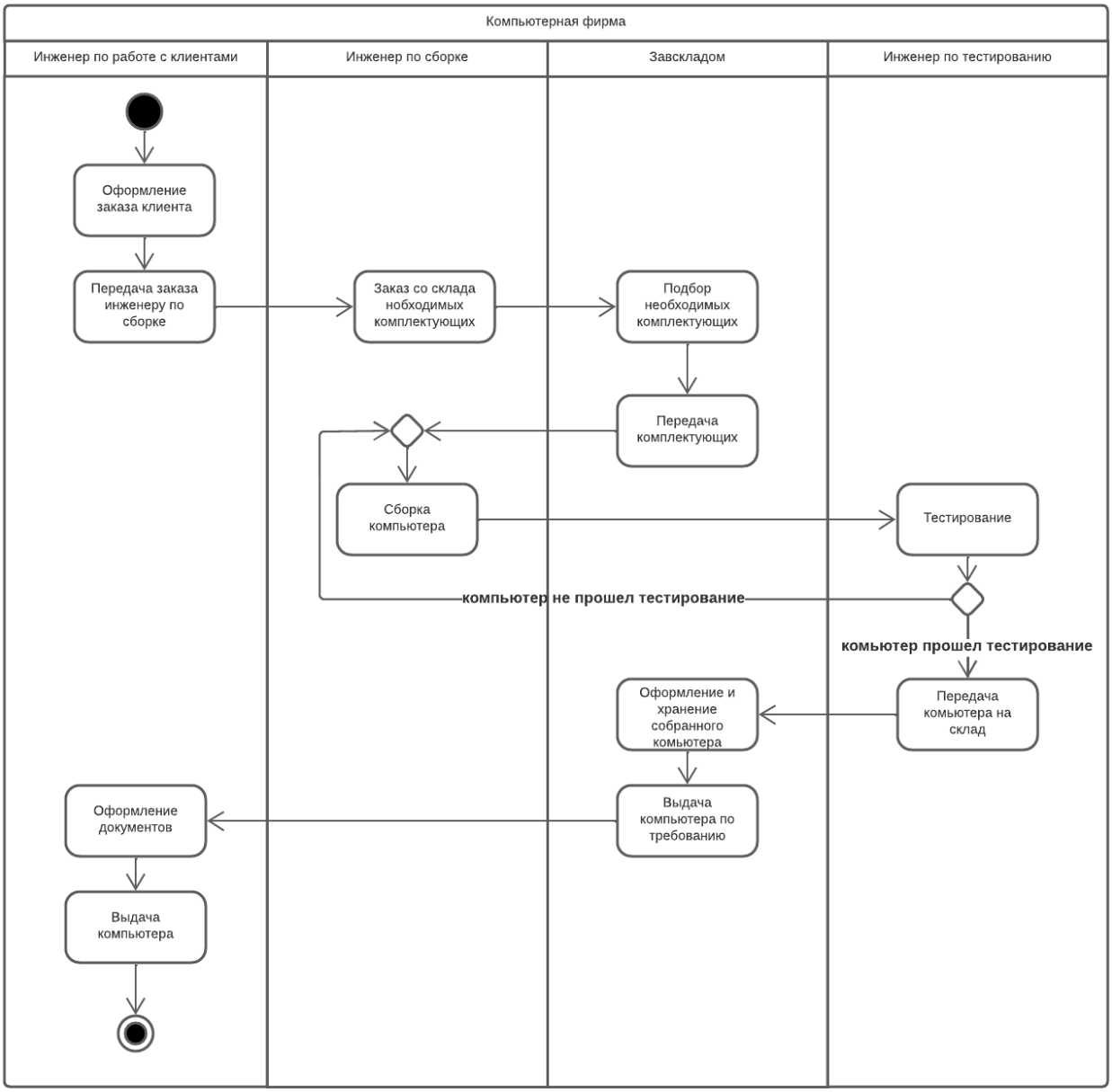


Рис.2. Диаграмма деятельности (активности)

Контрольные вопросы

7. Что собой представляет диаграмма деятельности (активности)?

8. Какие объекты применяются для построения диаграммы деятельности?

Список использованных источников

1. Зиновьев В.В. Моделирование процессов и систем : учебное пособие / Зиновьев В.В., Стародубов А.Н., Николаев П.И.. — Кемерово : Кузбасский государственный технический университет имени Т.Ф. Горбачёва, 2016. — 146 c. — ISBN 978-5-906888-10-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/109114.html

2. Лисяк В.В. Моделирование информационных систем : учебное пособие / Лисяк В.В., Лисяк Н.К.. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2018. — 88 c. — ISBN 978-5-9275-2881-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/87729.html

3. Чернышев А.Б. Теория информационных процессов и систем : учебное пособие / Чернышев А.Б., Антонов В.Ф., Суюнова Г.Б.. — Ставрополь : Северо-Кавказский федеральный университет, 2015. — 169 c. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/63140.html