МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Экономика и управление»

ОСНОВЫ ПОСТРОЕНИЯ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКИХ СИСТЕМ

Методические рекомендации к лабораторным работам для студентов специальности 1-25 01 07 «Экономика и управление на предприятии» очной и заочной форм обучения

Могилев 2020

УДК 004.94
ББК 32.973
О75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Экономика и управление» «30» апреля 2020 г., протокол № 9

Составитель ст. преподаватель Е. Г. Галкина

Рецензент канд. техн. наук, доц. Т. В. Пузанова

Методические рекомендации предназначены для студентов специальности 1-25 01 07 «Экономика и управление на предприятии» очной и заочной форм обучения.

Учебно-методическое издание

ОСНОВЫ ПОСТРОЕНИЯ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКИХ СИСТЕМ

Ответственный за выпуск	И. В. Ивановская
Корректор	Е. А. Галковская
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. .Уч.-изд. л. . Тираж 36 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№1/156 от 07.03.2019.
Пр–т Мира, 43, 212022, Могилев.

Содержание

Лабораторная работа № 1. Системный анализ и анализ требований к программной системе

Цель работы: изучить возможности методологии SADT. Научиться создавать модели бизнес-процессов, формулировать требования к проектируемой информационно-аналитической системе на основе анализа модели бизнес-процессов.

Задание к лабораторной работе.

Создать контекстную диаграмму в соответствии с семантикой предметной области индивидуального задания и выполнить её декомпозицию. Использовать не менее трёх уровней диаграмм.

Методические указания

Проектирование информационно–аналитической системы начинается с анализа требований, которым она должна удовлетворять. При объектно-ориентированном подходе анализ требований сводится к разработке моделей системы. Модели, разработанные и отлаженные в начале проекта, используются на последующих его этапах, облегчая программирование системы, её отладку и тестирование, сопровождение и дальнейшую модификацию.

Для того, чтобы провести анализ требований, следует подробно описать специфику экономической задачи, выбранной для автоматизации.

Для более наглядного представления этой информации необходимо построить модель бизнес-процессов. Наиболее удобным языком моделирования бизнес-процессов является IDEF0, в соответствии с которым система представляется как совокупность взаимодействующих работ или функций. Функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет чётко смоделировать логику и взаимодействие процессов организации.

Моделирование в IDEF0 начинается с создания контекстной диаграммы – диаграммы абстрактного уровня описания системы в целом, содержащей определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система; на определение субъекта влияют позиция, с которой рассматривается система, и цель моделирования – вопросы, на которые построенная модель должна дать ответ.

Сначала определяется область моделирования. В ходе моделирования область может корректироваться, но изначально она должна быть сформулирована, поскольку определяет направление моделирования. При формулировании области необходимо учитывать ширину и глубину моделирования. Ширина определяет, что будет рассматриваться внутри системы, а что снаружи. Глубина определяет, на каком уровне детализации модель является завершённой. При определении глубины системы необходимо помнить, что трудоёмкость построения модели растёт с увеличением глубины

декомпозиции. После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему.

Также перед началом моделирования необходимо чётко определить цель проектируемой системы и позицию (точку зрения), с которой наблюдается система и создаётся модель.

Под точкой зрения понимается перспектива, с которой наблюдалась система при построении модели. При построении модели учитываются мнения различных людей, но все они должны придерживаться единой точки зрения на модель. Точка зрения должна соответствовать цели и границам моделирования. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом. Методология SADT (акроним от англ. structured analysis and design technique) требует, чтобы модель всегда рассматривалась с одной позиции. Точка зрения диктует выбор нужной информации и форму её подачи.

IDEF0-модель предполагает наличие чётко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Конечным результатом является набор взаимоувязанных описаний, начиная с описания самого верхнего уровня всей системы и кончая подробным описанием деталей или операций системы. Такие описания называются диаграммами. IDEF0-модель объединяет и организует диаграммы в иерархические структуры, в которых диаграммы наверху модели являются более общими, чем детализированные диаграммы нижних уровней. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать следующие типы диаграмм:

- контекстную диаграмму (в модели она может быть лишь одна);
- диаграммы декомпозиции.

Контекстная диаграмма представляет собой общее описание системы и её взаимодействия с внешней средой. После описания системы в целом проводится разбиение её на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты указывают на соответствие реальных бизнес-процессов созданным диаграммам. После прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес–процессам на любом уровне модели.

Каждая IDEF0-диаграмма содержит блоки и дуги. Блоки соответствуют работам и обозначают процессы, функции или задачи, которые происходят в течение определённого времени и имеют распознаваемые результаты. Работы изображаются в виде прямоугольников. Все работы должны быть названы и определены. Имя работы выражается глаголом или отглагольным существительным, обозначающим действие (например, «Принять заказ», «Составление отчёта» и т. д.). IDEF0 требует, чтобы в диаграмме было

не более шести блоков. Это обеспечивает удобство чтения и понимания диаграмм. Каждая сторона блока имеет определённое назначение. Левая сторона блока предназначена для входов, верхняя – для управления, правая – для выходов, нижняя – для механизмов. Такое обозначение отражает определённые системные принципы: входы преобразуются в выходы, управление оговаривает условия выполнения преобразований, механизмы показывают, кто, что и как выполняет функцию.

Дуги на диаграмме IDEF0 изображаются одинарными линиями со стрелками на концах. Стрелки описывают взаимодействие работ, представляют собой некий объект или информацию и выражаются существительными (например, «Звонки клиентов», «Правила», «Бухгалтерская система»).

Между объектами и функциями возможны четыре отношения: вход, управление, выход, механизм. Каждое из них изображается дугой, связанной с определённой стороной блока.

Вход – материал или информация, которые используются или преобразуются работой для получения результата (выхода). Стрелка входа входит в левую сторону блока. Работа может не иметь стрелок входа. При описании технологических процессов не возникает проблем определения входов. При моделировании процессов обработки информации, когда стрелками являются данные, всё не так очевидно. Например, при выполнении функции «Приём пациента» карта пациента может быть на входе и на выходе, но качество этих данных меняется. Для того чтобы оправдать своё назначение, стрелки входа и выхода должны быть точно определены и указывать на то, что данные были переработаны (например, на выходе – «Заполненная карта пациента»). Часто сложно определить, являются ли данные входом или управлением. В этом случае помогает информация о том, изменяются данные в работе или нет. Если изменяются, то, скорее всего, это вход, если нет – управление.

Управление – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелка управления входит в верхнюю сторону блока. Управление влияет на работу, но не преобразуется ею. Если цель работы – изменить процедуру или стратегию, то эта процедура или стратегия будет для работы входом. В случае возникновения неопределённости в статусе стрелки (управление или вход) рекомендуется рисовать стрелку управления.

Выход – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Стрелка выхода исходит из правой стороны блока.

Механизм – ресурсы, которые выполняют работу, например, персонал предприятия, станки, ПО и т. д. Стрелка механизма входит в нижнюю сторону блока. По усмотрению механизмы могут не изображаться в модели.

Таким образом, IDEF0-диаграммы представляют входные–выходные преобразования и указывают правила этих преобразований. Дуги на них изображают интерфейсы между функциями системы, а также между системой и окружающей средой.

Стрелки на контекстной диаграмме служат для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у работы, или наоборот. Такие стрелки называются граничными. Пример контекстной диаграммы представлен на рисунке 1.

Рисунок 1 – Пример контекстной диаграммы

Диаграмма декомпозиции предназначена для детализации работы, следовательно, она содержит дочерние работы, имеющие общую родительскую работу. Нужно понимать, что работы нижнего уровня – это то же самое, что работы верхнего уровня, но в более детальном изложении. Следовательно, границы работы верхнего уровня – то же самое, что границы диаграммы декомпозиции. При декомпозиции работы входящие в неё и исходящие из неё стрелки должны обязательно присутствовать на диаграмме декомпозиции.

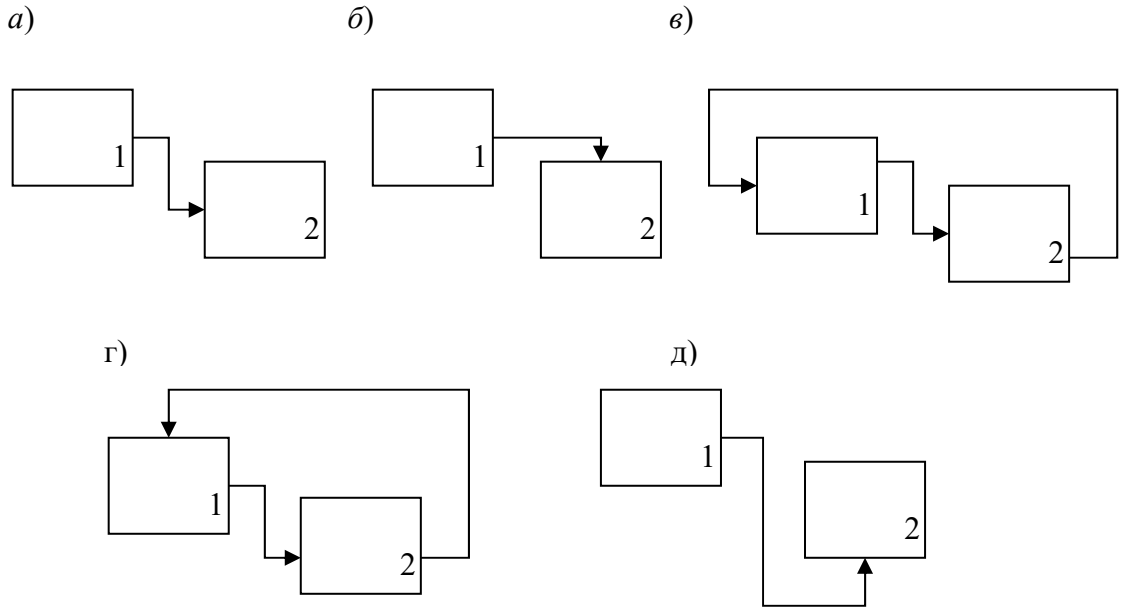
Допустимый интервал числа дочерних работ на диаграмме – от двух до восьми. Декомпозировать работу на одну работу не имеет смысла, а диаграммы с количеством работ более восьми плохо читаются. Для обеспечения наглядности и понимания моделируемых процессов рекомендуется использовать от трёх до шести блоков на одной диаграмме. Каждая из работ на диаграмме может быть декомпозирована. В левом верхнем углу блока изображается небольшая диагональная черта, которая показывает, что данная работа не была декомпозирована.

Работы на диаграммах располагаются по диагонали от левого верхнего угла к правому нижнему. Такой порядок называется порядком доминирования. Доминирование – это влияние, которое один блок оказывает на другие блоки диаграммы. В левом верхнем углу обычно помещается самая важная или выполняемая по времени первой работа. Вправо вниз располагаются менее важные или выполняемые позже работы. Такое размещение облегчает чтение диаграмм, кроме того, на нём основывается понятие взаимосвязей работ.

Блоки в IDEF0 должны быть пронумерованы. Блоки нумеруются слева направо. Номер работы показывается в правом нижнем углу. 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.

Для связи работ между собой используются внутренние стрелки, т. е. стрелки, которые начинаются у одной и кончаются у другой работы. В IDEF0 есть пять типов взаимосвязей между блоками: выход-вход, выход-управление, обратная связь по управлению, обратная связь по входу, выход-механизм. Примеры связей приведены на рисунке 2.



а – выход-вход; *б* – выход-управление; *в* – обратная связь по входу; *г* – обратная связь по управлению; д – выход-механизм

Рисунок 2 – Взаимосвязи между блоками

Связь выход-вход возникает, когда стрелка выхода вышестоящей работы направляется на вход нижестоящей.

Связь выход-управление появляется, когда выход вышестоящей работы направляется на управление нижестоящей. Связь по управлению показывает доминирование вышестоящей работы. Данные или объекты выхода вышестоящей работы не меняются в нижестоящей.

Обратная связь по входу используется, когда выход нижестоящей работы направляется на вход вышестоящей. Такая связь используется для описания циклов.

Обратная связь по управлению применяется, когда выход нижестоящей работы направляется на управление вышестоящей. Такая связь свидетельствует об эффективности бизнес-процессов.

Связь выход-механизм возникает, когда выход одной работы направляется на механизм другой. Эта взаимосвязь используется реже остальных и

показывает, что одна работа подготавливает ресурсы, необходимые для проведения другой работы.

Объекты, порождённые одной работой, могут использоваться сразу в нескольких других работах. Стрелки, порождённые в разных работах, могут представлять собой однородные объекты, применяемые в одном месте. Для моделирования таких ситуаций используются разветвляющиеся и сливающиеся стрелки. Дуги могут разветвляться и соединяться различными способами. Вся дуга или её часть может выходить из одного или нескольких блоков и заканчиваться в одном или нескольких блоках. Смысл разветвляющихся и сливающихся стрелок передаётся именованием каждой ветви стрелок. Существуют правила именования таких стрелок. Если стрелка именована до разветвления, а после разветвления ни одна из ветвей не именована, то подразумевается, что каждая ветвь моделирует те же данные или объекты, что и ветвь до разветвления.

Если стрелка именована до разветвления, а после разветвления какая–либо из ветвей тоже именована, то подразумевается, что эти ветви соответствуют именованию. Если при этом какая–либо ветвь после разветвления осталась неименованной, то подразумевается, что она моделирует те же данные или объекты, что и ветвь до разветвления. Недопустима ситуация, когда стрелка до разветвления не именована, а после разветвления не именована какая-либо из ветвей.

Правила именования сливающихся стрелок аналогичны.

Для изображения малозначимых объектов, которые не требуется изображать на всех уровнях диаграмм, может быть применён туннель. Выходом является туннелирование стрелки на самом нижнем уровне. Такое туннелирование называется «не-в-родительской-диаграмме».

Другим примером туннеля может быть ситуация, когда стрелка механизма мигрирует на нижний уровень, а механизм используется одинаково во всех работах. В этом случае стрелка на нижнем уровне удаляется, на родительской диаграмме она туннелируется, а в комментарии к стрелке можно указать, что механизм будет использоваться во всех работах дочерней диаграммы. Такое туннелирование называется «не-в-дочерней-работе».

Туннельная стрелка изображается с круглыми скобками на конце.

Пример диаграммы декомпозиции с ветвлением стрелок и туннелем представлен на рисунке 3.

Все работы модели нумеруются. Номер состоит из префикса и числа. Можно использовать любой префикс, но обычно используют префикс A. Контекстная работа имеет номер A0. Работы декомпозиции A0 имеют номера A1, A2 и т. д. Работы декомпозиции нижнего уровня имеют номер родительской работы и очередной порядковый номер, например, работы декомпозиции A3 будут иметь номера A31, A32, A33 и т. д. Работы образуют иерархию, где каждая работа может иметь одну родительскую и несколько дочерних работ, образуя дерево.

Работа в IDEF0 предполагает последовательное улучшение описания системы. Но процесс улучшения в некоторый момент должен быть прекращён.

Декомпозиция прекращается, когда диаграммы, образующие нижний уровень модели, достаточно детализированы для достижения цели.

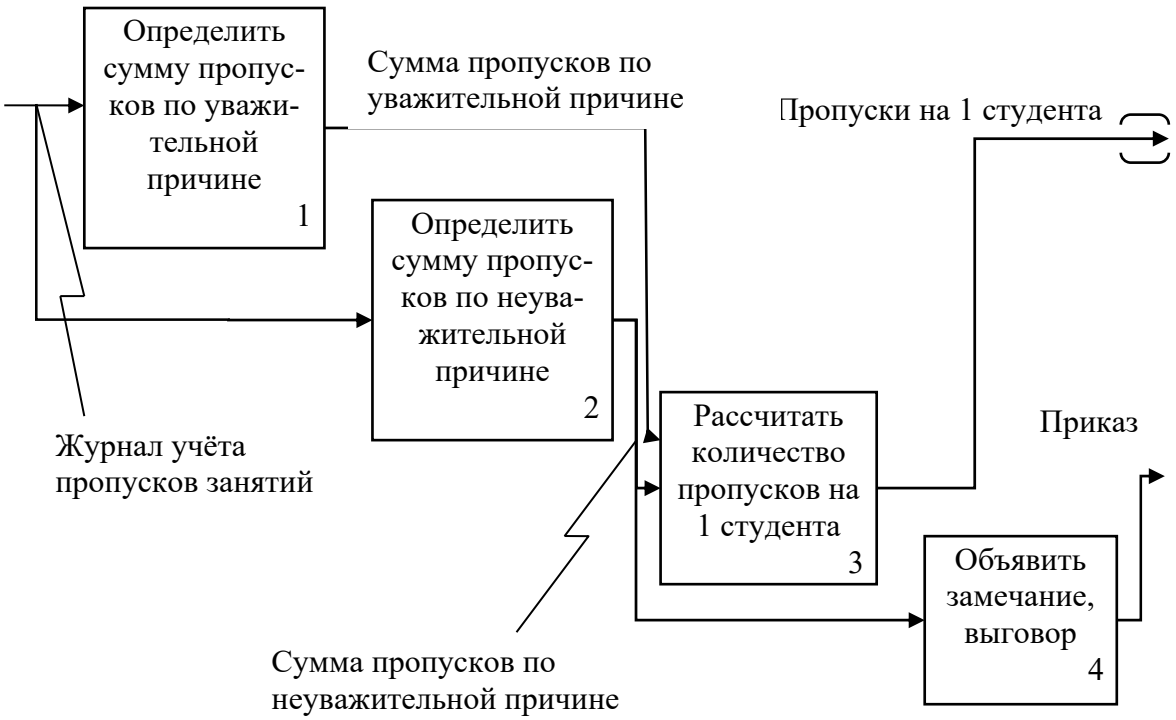


Рисунок 3 – Пример диаграммы декомпозиции

Контрольные вопросы

1 Что представляет собой модель в IDEF0?
2 Что обозначают работы в IDEF0?
3 Назовите правила наименования работ.
4 Каково назначение сторон прямоугольника работ на диаграммах?
5 Перечислите типы стрелок.
6 Назовите виды взаимосвязей.
7 Что называется граничными стрелками?

Лабораторная работа № 2. Создание контекстной диаграммы и диаграмм декомпозиции

Цель работы: освоить создание контекстной диаграммы и диаграмм декомпозиции в пакете AllFusion Process Modeler.

Задание к лабораторной работе.

Создать в пакете AllFusion Process Modeler полученную в ходе выполнения лабораторной работы №1 контекстную диаграмму.

Методические указания

В качестве примера рассмотрим деятельность вымышленной компании. Компания занимается в основном сборкой и продажей настольных компьютеров и ноутбуков. Компания не производит компоненты самостоятельно, а только собирает и тестирует компьютеры.

Основные процедуры в компании таковы:

– продавцы принимают заказы клиентов;

– операторы группируют заказы по типам компьютеров;

– операторы собирают и тестируют компьютеры;

– операторы упаковывают компьютеры согласно заказам;

– кладовщик отгружает клиентам заказы.

Компания использует купленную бухгалтерскую информационную систему, которая позволяет оформить заказ, счет и отследить платежи по счетам.

Запустите AllFusion Process Modeler.

Если появляется диалог ModelMart Connection Manager, нажмите на кнопку Cancel.

Щелкните по кнопке  . Появляется диалог I would like to. Внесите имя модели ***«Деятельность компании»*** и выберите Type – IDEF0. Нажмите ОК. В появившемся окне Properties for New Models внесите имя автора.

Автоматически создается контекстная диаграмма.

Обратите внимание на кнопку –  : на панели инструментов. Эта кнопка включает и выключает инструмент просмотра и навигации – Model Explorer (появляется слева). Model Explorer имеет три вкладки – Activities, Diagrams и Objects. Во вкладке Activities щелчок правой кнопкой по объекту позволяет редактировать его свойства.

Если вам непонятно, как выполнить то или иное действие, вы можете вызвать помощь – клавиша F1 или меню Help.

Перейдите в меню Model/Model Properties. Во вкладке General диалога Model Properties следует внести имя модели ***«Деятельность компании»***, имя проекта ***«Модель деятельности компании»***, имя автора и тип модели – Time Frame: AS-IS.

Во вкладке Purpose внесите цель – «Purpose: Моделировать текущие (AS-IS) бизнес-процессы компании» и точку зрения – «Viewpoint: Директор».

Во вкладке Definition внесите определение «Это учебная модель, описывающая деятельность компании» и цель «Scope: Общее управление бизнесом компании: исследование рынка, закупка компонентов, сборка, тестирование и продажа продуктов».

Перейдите на контекстную диаграмму и правой кнопкой мыши щелкните по работе. В контекстном меню выберите Name. Во вкладке Name внесите имя «***Деятельность компании***».

Во вкладке Definition внесите определение «***Текущие бизнес-процессы компании***».

Создайте стрелки на контекстной диаграмме (таблица 1).

Таблица 1 – Стрелки контекстной диаграммы

Arrow Name	Arrow Definition	Arrow Type
Бухгалтерская система	Оформление счетов, оплата счетов, работа с заказами	Mechanism
Звонки клиентов	Запросы информации, заказы, техподдержка и т. д.	Input
Правила и процедуры	Правила продаж, инструкции по сборке, процедуры тестирования, критерии производительности и т. д.	Control
Проданные продукты	Настольные и портативные компьютеры	Output

С помощью кнопки **T** внесите текст в поле диаграммы – точку зрения и цель. Результат выполнения показан на рисунке 4.

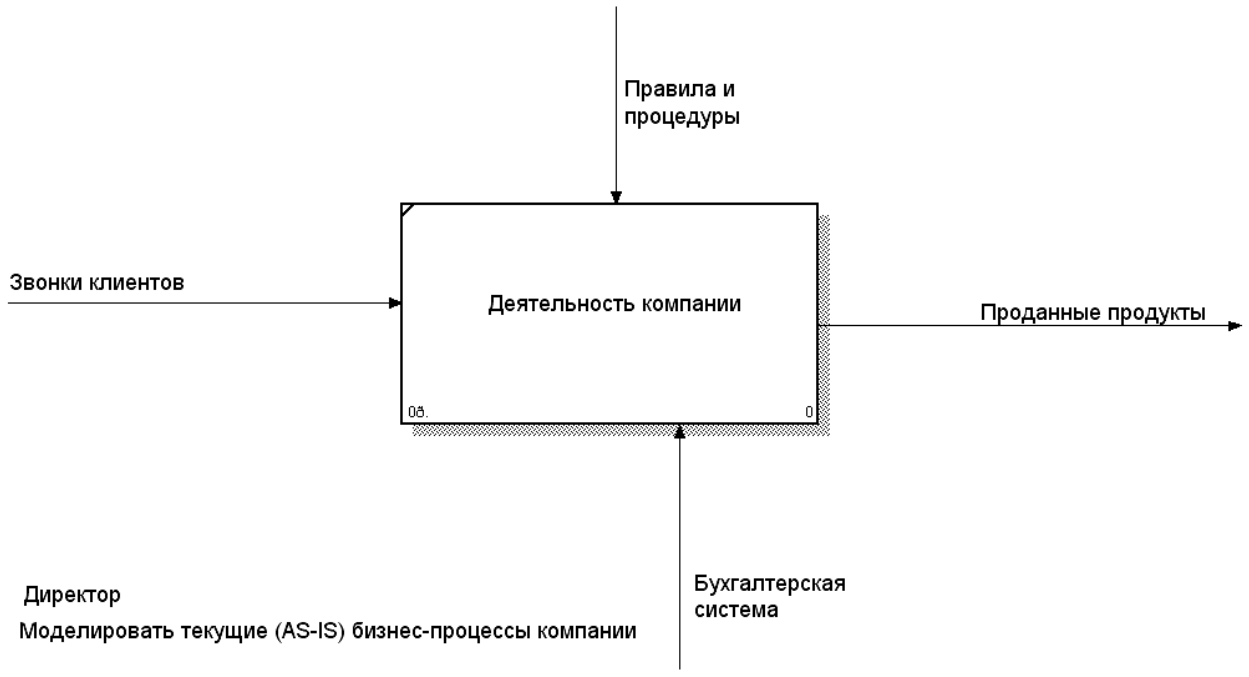


Рисунок 4 – Контекстная диаграмма

Выберите кнопку перехода на нижний уровень ▼ в палитре инструментов и в диалоге Activity Box Count установите число работ на диаграмме нижнего уровня – 3 – и нажмите ОК (рисунок 5).

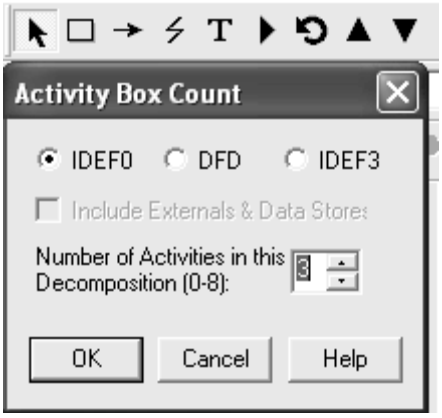


Рисунок 5 – Диалог Activity Box Count

Автоматически будет создана диаграмма декомпозиции. Правой кнопкой мыши щелкните по работе, выберите Name и внесите имя работы. Повторите операцию для всех трех работ. Затем внесите определение, статус и источник для каждой работы согласно таблицы 2.

Таблица 2 – Работы диаграммы декомпозиции А0

Activity Name	Definition
Продажи и маркетинг	Телемаркетинг и презентации, выставки
Сборка и тестирование компьютеров	Сборка и тестирование настольных и портативных компьютеров
Отгрузка и получение	Отгрузка заказов клиентам и получение компонентов от поставщиков

Для изменения свойств работ после их внесения в диаграмму можно воспользоваться словарем работ. Вызов словаря – меню Dictionary/Activity (рисунок 6).

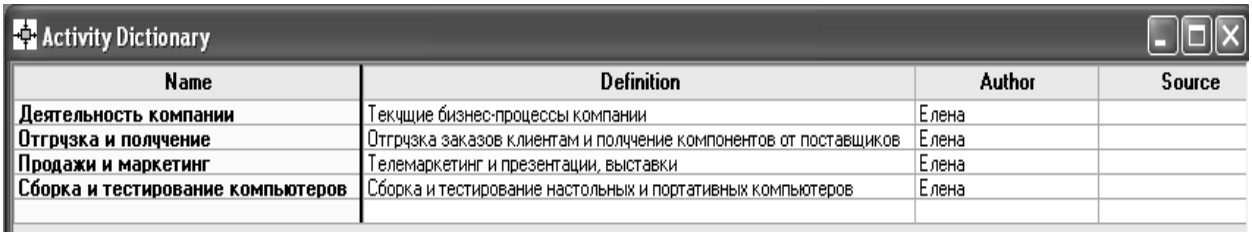


Рисунок 6 – Словарь Activity Dictionary

Если описать имя и свойства работы в словаре, ее можно будет внести в диаграмму позже с помощью кнопки в палитре инструментов.

Невозможно удалить работу из словаря, если она используется на какой-либо диаграмме. Если работа удаляется из диаграммы, из словаря она не удаляется. Имя и описание такой работы могут быть использованы в дальнейшем. Для добавления работы в словарь необходимо перейти в конец списка и щелкнуть правой кнопкой по последней строке. Возникает новая строка, в которой нужно внести имя и свойства работы. Для удаления всех имен работ, не использующихся в модели, щелкните по кнопке (Purge).

Перейдите в режим рисования стрелок. Свяжите граничные стрелки (кнопка на палитре инструментов так, как показано на рисунке 7).

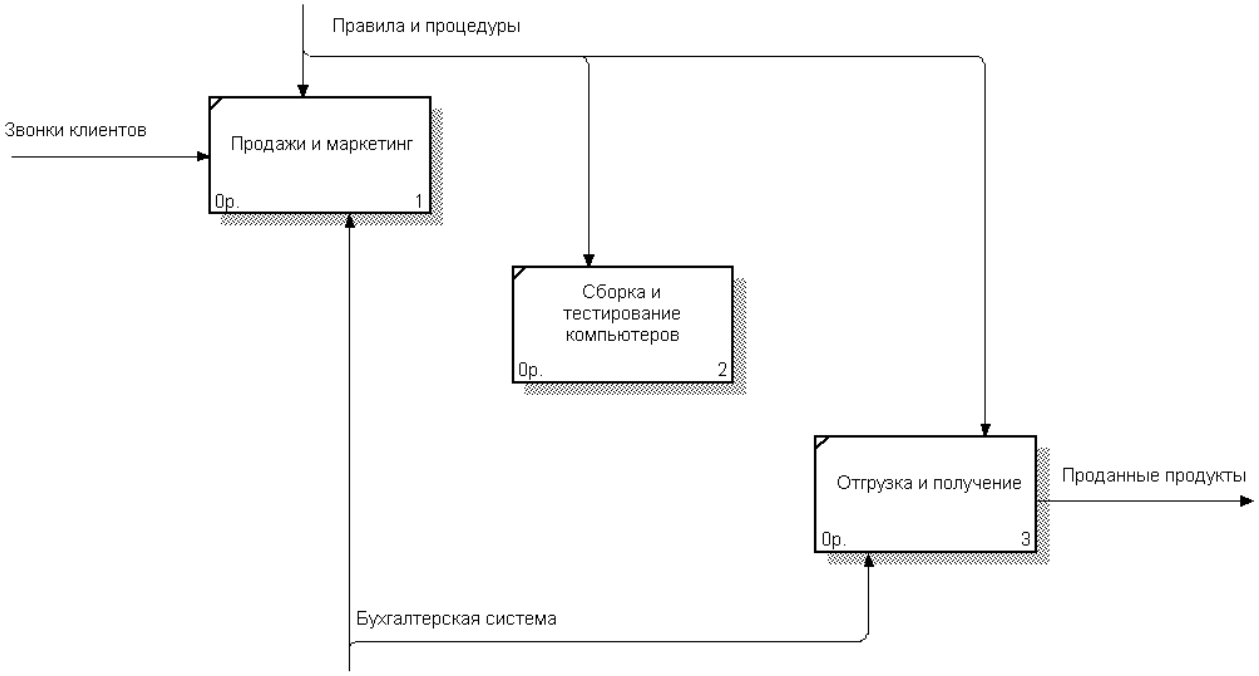


Рисунок 7 – Связанные граничные стрелки на диаграмме A0

Правой кнопкой мыши щелкните по ветви стрелки управления работы ***«Сборка и тестирование компьютеров»*** и переименуйте ее в ***«Правила сборки и тестирования»*** (рисунок 8).

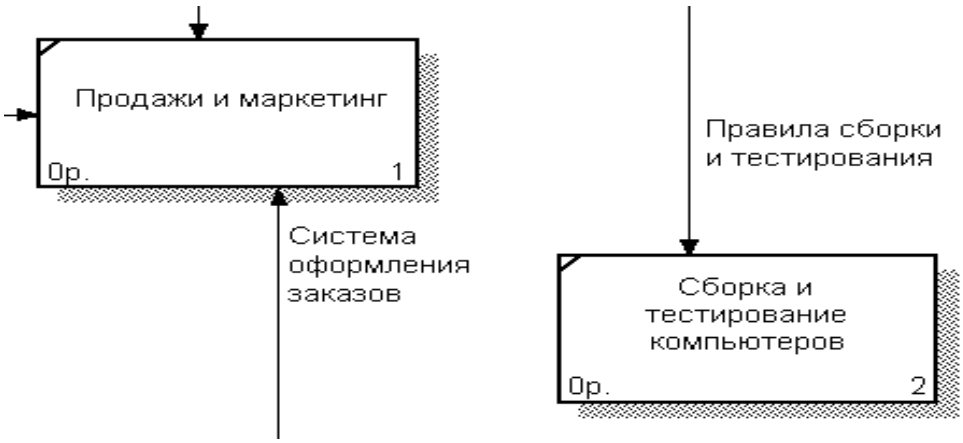


Рисунок 8 – Стрелка «Правила сборки и тестирования»

Внесите определение для новой ветви: «Инструкции по сборке, процедуры тестирования, критерии производительности и т. д.» Правой кнопкой мыши щелкните по ветви стрелки механизма работы «***Продажи и маркетинг***» и переименуйте ее в «***Систему оформления заказов***».

Альтернативный метод внесения имен и свойств стрелок – использование словаря стрелок (вызов словаря – меню Dictionary/Arrow).

Если внести имя и свойства стрелки в словарь, ее можно будет внести в диаграмму позже. Стрелку нельзя удалить из словаря, если она используется на какой-либо диаграмме. Если удалить стрелку из диаграммы, из словаря она не удаляется. Имя и описание такой стрелки может быть использовано в дальнейшем. Для добавления стрелки необходимо перейти в конец списка и щелкнуть правой кнопкой по послед ней строке. Возникает новая строка, в которой нужно внести имя и свойства стрелки.

Создайте новые внутренние стрелки так, как показано на рисунке 9.

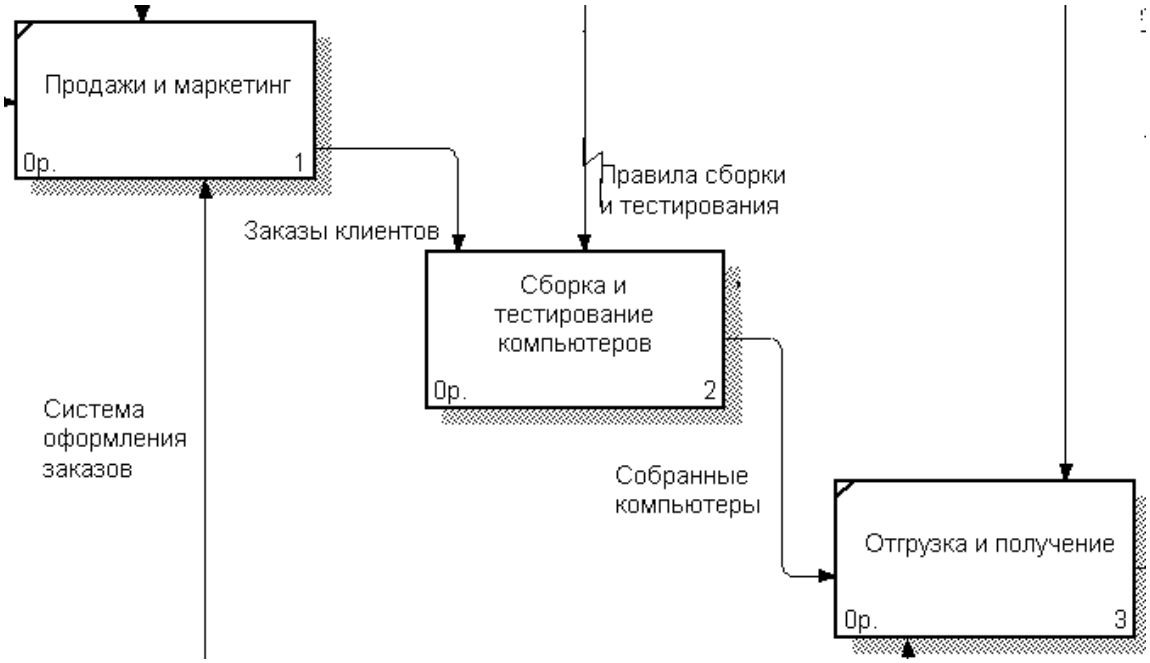


Рисунок 9 – Внутренние стрелки диаграммы A0

Создайте стрелку обратной связи (по управлению) «***Результаты сборки и тестирования***», идущую от работы «***Сборка и тестирование компьютеров***» к работе «***Продажи и маркетинг***». Измените стиль стрелки (толщина линий) и установите опцию Extra Arrowhead (из контекстного меню). Методом drag&drop перенесите имена стрелок так, чтобы их было удобнее читать. Если необходимо, установите Squiggle (из контекстного меню). Результат изменений показан на рисунке 10.

Создайте новую граничную стрелку выхода «***Маркетинговые материалы***», выходящую из работы «***Продажи и маркетинг***». Эта стрелка автоматически не попадает на диаграмму верхнего уровня и имеет квадратные скобки на наконечнике . Щелкните правой кнопкой мыши по квадратным

скобкам и выберите пункт меню Arrow Tunnel. В диалоге Border Arrow Editor выберите опцию Resolve it to Border Arrow. Для стрелки ***«Маркетинговые материалы»*** выберите опцию Trim из контекстного меню. Результат выполнения упражнения 2 показан на рисунке 11.

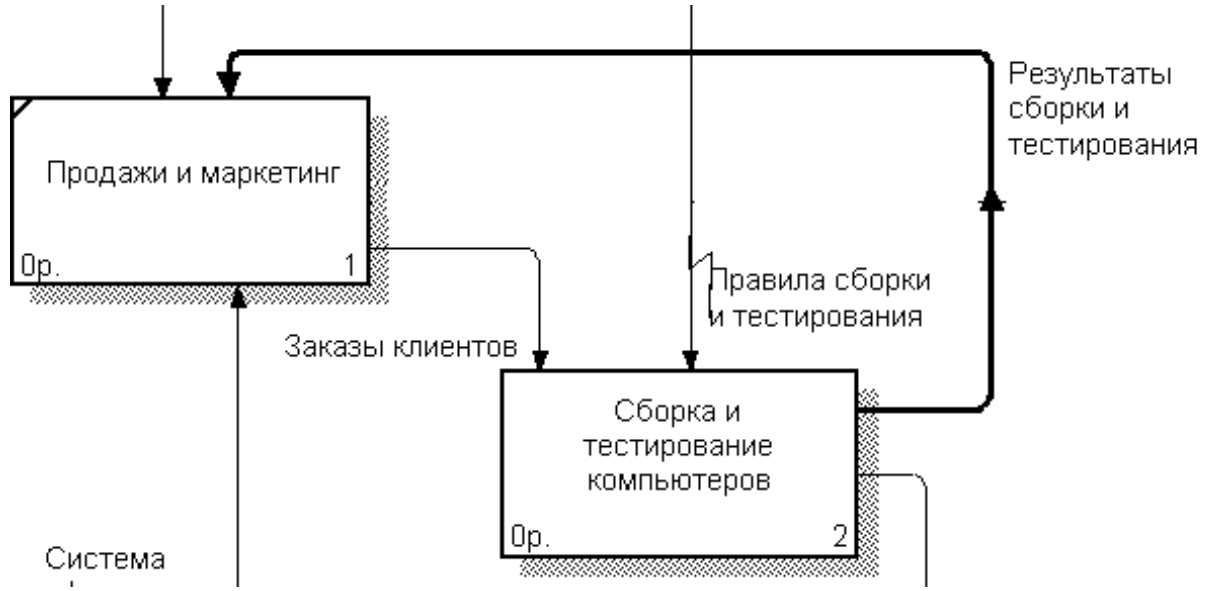


Рисунок 10 – Результат редактирования стрелок на диаграмме A0

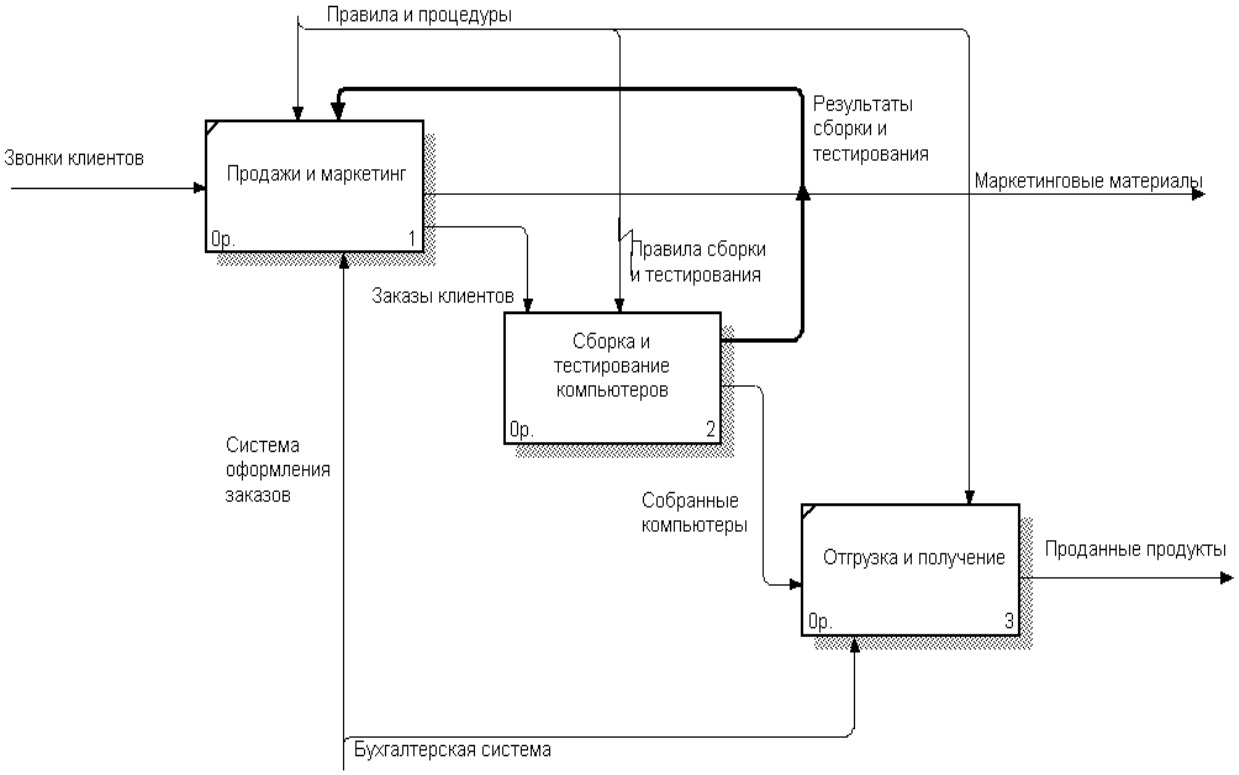


Рисунок 11 – Результат выполнения

Контрольные вопросы

1 Какие методологии поддерживает AllFusion Process Modeler?
2 Перечислите основные элементы главного окна AllFusion Process Modeler.
3 Опишите процесс создания новой модели в AllFusion Process Modeler.
4 Как провести связь между работами?
5 Как создать имя работы?
6 Опишите процесс декомпозиции работы.
7 Как добавить работу на диаграмму?
8 Как разрешить туннелированные стрелки?
9 Может ли модель AllFusion Process Modeler содержать диаграммы нескольких методологий?

Лабораторная работа № 3. Создание диаграммы дерева узлов и FEO-диаграммы

Цель работы: освоить создание диаграммы дерева узлов и FEO-диаграммы.

Задание к лабораторной работе.
Представить созданную в лабораторной работе № 2 диаграмму в виде дерева узлов и для выбранного фрагмента создать FEO-диаграмму.

Методические указания

Диаграмма деревьев узлов показывает иерархию работ в модели и позволяет рассмотреть всю модель целиком, но не демонстрирует взаимосвязи между работами (рисунок 12). Процесс создания модели работ является итерационным, а работы могут менять своё расположение в дереве узлов. Чтобы не запутаться и проверить способ декомпозиции, следует после каждого изменения создавать диаграмму дерева узлов.

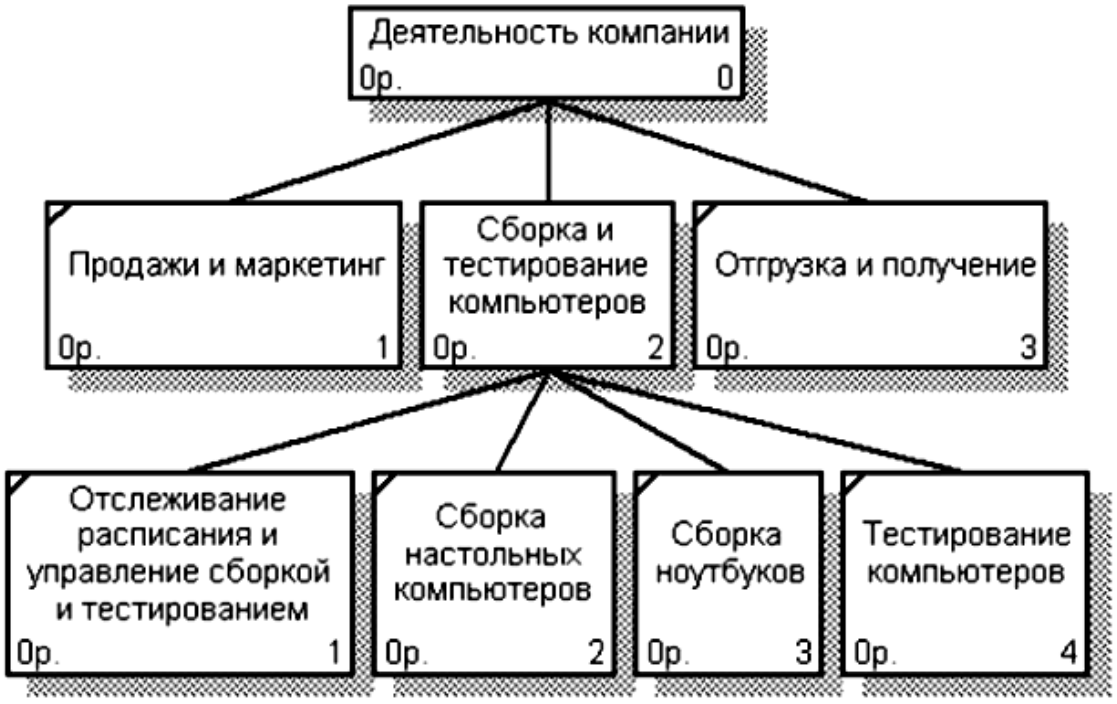


Рисунок 12 – Диаграмма дерева узлов

Для создания диаграммы дерева узлов необходимо выбрать в меню пункт Diagram/Add Node Tree. Возникает диалог Node Tree Definition. В нём следует указать глубину дерева – Number of Levels (по умолчанию 3) и корень дерева (по умолчанию – родительская работа текущей диаграммы). По умолчанию нижний уровень декомпозиции показывается в виде списка, остальные работы – в виде прямоугольников. Для отображения всего дерева в виде прямоугольников следует выключить опцию Bullet Last Level. При создании дерева узлов необходимо указать имя диаграммы, поскольку если в нескольких диаграммах в качестве корня на дереве узлов использовать одну и ту же работу, все эти диаграммы получат одинаковый номер (номер узла + постфикс N, например, AON) и в списке открытых диаграмм (пункт меню Window) их можно будет различить только по имени.

Диаграммы «только для экспозиции» (FEO) часто используются в модели для иллюстрации других точек зрения, для отображения отдельных деталей, которые не поддерживаются явно синтаксисом IDEF0. Диаграммы FEO позволяют нарушить любое синтаксическое правило, поскольку являются просто картинками и не включаются в анализ синтаксиса. Для создания диаграммы FEO следует выбрать пункт меню Diagram/Add FEO Diagram. В возникающем диалоге Add New FEO Diagram нужно указать имя диаграммы FEO и тип родительской диаграммы. Новая диаграмма получает номер, который генерируется автоматически (номер родительской диаграммы по узлу + постфикс F, например A1F).

Граничные рамки диаграммы называются каркасом диаграммы. Каркас содержит заголовок (верхняя часть рамки) и подвал (нижняя часть). Заголовок каркаса используется для отслеживания диаграммы в процессе моделирования. Нижняя часть используется для идентификации и позиционирования в

иерархии диаграммы. Смысл элементов каркаса приведен в таблицах 3 и 4. Значения полей каркаса задаются в диалоге Diagram Properties.

Таблица 3 – Поля заголовка каркаса (слева направо)

Поле	Назначение и содержание
Used At	Используется для указания на родительскую работу в случае, если на текущую диаграмму ссылались посредством стрелки вызова
Autor, Date, Rev, Project	Имя создателя, дата создания и имя проекта, в рамках которого была создана диаграмма. REV-дата последнего редактирования диаграммы
Notes	Используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии) указать число замечаний, вычёркивая цифру из списка каждый раз при внесении нового замечания
Status	Стадия создания диаграммы, отображая все этапы публикации
Working	Новая диаграмма, обновлённая диаграмма или новый автор диаграммы
Draft	Диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению
Recommended	Диаграмма и все её сопровождающие документы прошли экспертизу. Новых изменений не ожидается
Publication	Диаграмма готова к окончательной печати и публикации
Reader	Имя читателя (эксперта)
Date	Дата прочтения (экспертизы)
Context	Схема расположения работ в диаграмме верхнего уровня. Родительская работа показана тёмным прямоугольником, остальные – светлым. На контекстной диаграмме (A-0) показана надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы

Таблица 4 – Поля подвала каркаса (слева направо)

Поле	Назначение и содержание
Node	Номер узла диаграммы (номер родительской работы)
Title	Имя диаграммы. По умолчанию — имя родительской работы
Number	C-Number, уникальный номер версии диаграммы
Page	Номер страницы, может использоваться при формировании папки

Контрольные вопросы

1 Что отражает диаграмма деревьев узлов?

2 Что отражает FEO диаграмма?

3 Почему FEO диаграммы позволяют нарушить любое синтаксическое правило?

4 Как присваивается номер FEO диаграммы?

Лабораторная работа № 4. Расщепление и слияние моделей

Цель работы: изучить возможность коллективной работы над проектом посредством расщепления и слияния моделей.

Задание к лабораторной работе.

На примере индивидуального задания выполнить расщепление и слияние моделей.

Методические указания

Возможность слияния и расщепления моделей обеспечивает коллективную работу над проектом. Так, руководитель проекта может создать декомпозицию верхнего уровня и дать задание аналитикам продолжить декомпозицию каждой ветви дерева в виде отдельных моделей. После окончания работы над отдельными ветвями все подмодели могут быть слиты в единую модель. С другой стороны, отдельная ветвь модели может быть отщеплена для использования в качестве независимой модели, для доработки или архивирования.

AllFusion Process Modeler использует для слияния и разветвления моделей стрелки вызова. Для слияния необходимо выполнить следующие условия:

- обе сливаемые модели должны быть открыты;
- имя модели-источника, которое присоединяют к модели-цели, должно совпадать с именем стрелки вызова работы в модели-цели;
- стрелка вызова должна исходить из недекомпозируемой работы;
- имена контекстной работы модели-источника и работы на модели-цели, к которой подсоединяют модель-источник, должны совпадать;
- модель-источник должна иметь хотя бы одну диаграмму декомпозиции.

Для слияния моделей нужно щёлкнуть правой кнопкой по работе со стрелкой вызова в модели-цели и в меню выбрать пункт Merge Model. Появляется диалог, в котором указывают опции слияния. При слиянии моделей объединяются словари стрелок и работ. В случае одинаковых определений возможна их перезапись или принятие из модели-источника. То же относится к именам стрелок, хранилищам данных и внешним ссылкам.

После подтверждения слияния модель-источник подсоединяется к модели-цели, стрелка вызова исчезает, а работа, от которой отходила стрелка вызова, становится декомпозируемой – к ней подсоединяется диаграмма декомпозиции первого уровня модели-источника. Стрелки, касающиеся работы на диаграмме модели-цели, автоматически не мигрируют в декомпозицию, а отображаются как неразрешённые. Их следует туннелировать вручную.

В процессе слияния модель-источник остается неизменной и к модели-цели подключается её копия. Если затем модель-источник будет редактироваться, эти изменения не попадут в ветви модели-цели.

Разделение моделей производится аналогично. Для отщепления ветви от модели следует щёлкнуть правой кнопкой мыши по декомпозированной работе

и выбрать в меню пункт Split Model. В появившемся диалоге Split Options следует указать имя создаваемой модели. После подтверждения расщепления в старой модели работа станет недекомпозированной, будет создана стрелка вызова, её имя будет совпадать с именем новой модели и, наконец, будет создана новая модель, причём имя контекстной работы будет совпадать с именем работы, от которой была «оторвана» декомпозиция.

Контрольные вопросы

1 Назовите причины расщепления модели.
2 Как в моделировании в нотации IDEF0 отражается расщепление модели?
3 Как в моделировании в нотации IDEF0 отражается слияние модели?
4 Какие могут быть изменения в исходной модели после ее слияния?
5 Какие ошибки могут возникнуть при слиянии моделей?

Лабораторная работа № 5. Создание диаграммы IDEF3

Цель работы: освоить методику создания диаграммы IDEF3.

Задание к лабораторной работе.
На примере индивидуального задания создать диаграмму IDEF3.

Методические указания

IDEF3 (workflow diagramming) – методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы IDEF3 могут быть применены в моделировании бизнес-процессов для анализа завершённости процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

IDEF3 дополняет IDEF0 и содержит всё необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Каждая работа в IDEF3 описывает какой-либо сценарий бизнес–процесса и может являться составляющей другой работы. Работы именуются отглагольным существительным, обозначающим процесс действия.

Единицы работы (Unit of Work, UOW) являются центральными компонентами модели. Они изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным существительным, обозначающим процесс действия, и номер (идентификатор); другое существительное в составе той же фразы отображает основной результат работы (например,

«Изготовление изделия»). Часто существительное меняется в процессе моделирования, поскольку модель может уточняться и редактироваться. Идентификатор работы присваивается при создании и не меняется никогда. Даже если работа будет удалена, её идентификатор не будет использоваться для других работ. Обычно номер работы состоит из номера родительской работы и порядкового номера на текущей диаграмме.

Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи, стиль которых устанавливается через меню Edit/Arrow Style:

– временное предшествование (Precedence) – сплошная линия, связывающая единицы работ (UOW). Рисуется слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнётся;

– отношение (Relational Link) – пунктирная линия, использующаяся для изображения связей между единицами работ (UOW), а также между единицами работ и объектами ссылок. Она показывает, что работа-источник не обязательно должна закончиться прежде, чем работа-цель начнётся. Более того, работа-цель может закончиться прежде, чем закончится работа-источник;

– поток объектов (Object Flow) – стрелка с двумя наконечниками. 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 началом нескольких работ или одна работа может ожидать окончания нескольких работ. Для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения событий, которые должны быть завершены перед началом следующей работы, используются перекрёстки (Junction). Различают перекрёстки для слияния (Fan-in Junction) и разветвления стрелок (Fan-out Junction). Перекрёсток не может использоваться одновременно для слияния и для разветвления. Для внесения перекрёстка служит кнопка Junction в палитре инструментов. В диалоге Select Junction Type указывают тип перекрёстка. Смысл каждого типа приведён в таблице 5.

Все перекрёстки на диаграмме нумеруются, каждый номер имеет префикс J. Можно редактировать свойства перекрёстка при помощи диалога Junction Properties, который вызывается в контекстном меню перекрёстка командой Definition/Note. В отличие от IDEF0 и DFD в IDEF3 стрелки могут сливаться и разветвляться только через перекрёстки.

Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрёстком или работой. Для внесения объекта ссылки служит кнопка Referent в палитре инструментов. Объект ссылки изображается в виде прямоугольника. Имя его задаётся в диалоге Referent (пункт Name). В качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты

ссылки должны быть связаны с единицами работ перекрёстками или пунктирными линиями. При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки. Типы объектов ссылок приведены в таблице 6.

Таблица 5 – Типы перекрёстков

Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
Synchronous OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Таблица 6 – Типы объектов ссылок

Тип объекта ссылки	Цель описания
OBJECT	Описывает участие важного объекта в работе
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ). Если все работы цикла присутствуют на текущей диаграмме, цикл может изображаться стрелкой, возвращающейся на стартовую работу. GOTO может ссылаться на перекрёсток
UOB (Unit of behaviour)	Применяется, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла. Например, работа «Контроль качества» может быть использована в процессе «Изготовление изделия» несколько раз, после каждой операции
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. NOTE является альтернативой внесению текстового объекта в диаграмму
ELAB (Elaboration)	Используется для усовершенствования графиков или их более детального описания, обычно для разветвления и слияния стрелок на перекрёстках

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 состоит из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме.

В результате дополнения диаграмм IDEF0 диаграммами DFD и IDEF3 может быть создана смешанная модель, которая наилучшим образом описывает все стороны деятельности предприятия. Иерархию работ в смешанной модели можно увидеть в окне Model Explorer. Модели в нотации IDEF0 изображаются зелёным цветом, в IDEF3 – жёлтым, в DFD – голубым.

Контрольные вопросы

1 Что описывает диаграмма IDEF3?
2 Перечислите составные элементы диаграмм IDEF3.
3 Что называется внешней сущностью?
4 Что описывают хранилища?
5 Что показывают связи в диаграммах IDEF3?
6 Перечислите типы стрелок в диаграммах IDEF3.
7 Что называется перекрестком?
8 Кратко опишите типы перекрестков в диаграммах IDEF3.
9 Как осуществляется идентификация перекрестков в диаграммах IDEF3?
10 Как отразить иерархию работ в смешанной модели?
11 Чем отличается изображение IDEF0, IDEF3 и DFD в смешанных моделях?
12 В чем отличие стрелок в IDEF3 от IDEF0 и DFD?

Лабораторная работа № 6. Стоимостной анализ в программе структурного моделирования

Цель работы: освоить методику проведения функционально-стоимостного анализа в пакете AllFusion Process Modeler.

Задание к лабораторной работе.

Внести данные о стоимости работ и ресурсов в соответствии с семантикой индивидуального задания и определить стоимость различных процессов.

Методические указания

После построения модели проводится анализ бизнес–процессов, потоки данных и объектов улучшаются, а в результате строится новая модель. Как правило, строится несколько моделей, из которых по какому–либо критерию выбирается наилучшая.

AllFusion Process Modeler предоставляет два инструмента для оценки модели – функционально–стоимостный анализ (Activity Based Costing, ABC) и свойства, определяемые пользователем (User Defined Properties, UDP).

ABC – это технология выявления и исследования стоимости той или иной функции. Исходными данными для него являются затраты на ресурсы (материалы, персонал и т. д.). Обычно ABC применяется для того, чтобы понять происхождение затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Reengineering, BPR).

ABC может проводиться только когда модель работы последовательная (следует синтаксическим правилам IDEF0), корректная (отражает бизнес), полная (охватывает всю рассматриваемую область) и стабильная (проходит экспертизу без изменений), т. е. когда создание модели работы закончено.

ABC включает следующие основные понятия:

– объект затрат — причина, по которой работа выполняется, обычно основной выход работы. Стоимость работ есть суммарная стоимость объектов затрат («Сборка и тестирование компьютеров»);

– источник затрат (cost driver) – характеристики входов и управлений работы («Заказы клиентов», «Правила сборки», «Персонал производственного отдела»), которые влияют на то, как выполняется и как долго длится работа;

– центры затрат (cost centers), которые соответствуют статьям расхода.

При проведении ABC сначала задаются единицы измерения времени и денег. Для этого следует вызвать диалог Model Properties (меню Model), закладка ABC Units. Если в списке выбора отсутствует необходимая валюта, её можно добавить. Диапазон измерения времени в списке Unit of measurment достаточен для большинства случаев – от секунд до лет.

Затем описываются центры затрат, для внесения которых необходимо вызвать диалог Cost Center Editor из меню Model. Каждому центру затрат следует дать описание в окне Definition. Список центров затрат упорядочен. Порядок в списке можно менять.

Для задания стоимости работы следует щёлкнуть правой кнопкой мыши по работе и в меню выбрать Cost. В диалоге Activity Cost указываются частота проведения данной работы в рамках процесса (Frequency) и продолжительность (Duration). Затем следует выбрать в списке один из центров затрат и в окне Cost задать его стоимость. Аналогично назначаются суммы по каждому центру затрат, т. е. задаётся стоимость каждой работы по каждой статье расхода. Если в процессе назначения стоимости требуется внести дополнительные центры затрат, диалог Cost Center Editor вызывается прямо из диалога Activity Properties/Cost соответствующей кнопкой.

Общие затраты по работе рассчитываются как сумма по всем центрам затрат. При вычислении затрат вышестоящей работы сначала вычисляется произведение затрат дочерней работы на частоту работы, затем результаты складываются. Если во всех работах модели включен режим Compute from Decompositions, вычисления автоматически проводятся по всей иерархии работ.

Этот упрощённый принцип подсчёта справедлив, если работы выполняются последовательно. Встроенные возможности позволяют разрабатывать упрощённые модели стоимости, которые оказываются чрезвычайно полезными при предварительной оценке затрат. Если схема выполнения более сложная

(например, работы производятся альтернативно), можно отказаться от подсчёта и задать итоговые суммы для каждой работы вручную (Override Decompositions). В этом случае результаты расчётов с нижних уровней декомпозиции будут игнорироваться и при расчётах на верхних уровнях будет учитываться сумма, заданная вручную. На любом уровне результаты расчётов сохраняются, поэтому при выключении опции Override Decompositions расчёт снизу-вверх производится обычным образом.

Результаты ABC могут повлиять на очередность выполнения работ. Результаты ABC представляются в отчёте, настройка которого проводится в диалоговом окне Activity Cost Report (меню Tools/Reports/Activity Cost Report). Отчёт позволяет документировать имя, номер, определение и стоимость работ как суммарно, так и раздельно по центрам затрат.

Результаты отображаются и на диаграммах. В левом нижнем углу работы показывается или стоимость (по умолчанию), или продол-жительность, или частота проведения работы. Настройка осуществляется в диалоге Model Properties (меню Model/Model Properties), закладка Display (ABC Data, ABC Units).

Контрольные вопросы

1 Поясните назначение ABC анализа.
2 Что такое центр затрат?
3 Что такое источник затрат?
4 Что такое объект затрат?
5 Опишите последовательность назначения стоимости выполнения работ для разных уровней модели.
6 Как на диаграммах IDEF0 осуществляется назначение стоимости каждой работы?
7 Как рассчитываются общие затраты по работе?
8 Чем отличается стоимость работы верхнего уровня от стоимости работ нижнего уровня?

Лабораторная работа № 7. Использование категорий UDP

Цель работы: изучить правила создания и применения категорий UDP.

Задание к лабораторной работе.
На примере индивидуального задания, используя категории UDP, внесите описание работ.

Методические указания

ABC позволяет оценить стоимостные характеристики системы. Если этого недостаточно, есть возможность внесения собственных метрик – свойств,

определенных пользователем, – (User Defined Properties, UDP). UDP позволяют провести дополнительный анализ, но без суммирующих подсчётов.

Для описания UDP служит диалог User-Defined Property Editor (меню Model/UDP Definition Editor). В верхнем окне диалога вносится имя UDP, в списке выбора Datatype описывается тип свойства. Имеется возможность задания 18 различных типов UDP, в том числе управляющих команд и массивов, объединённых по категориям. Для внесения категории следует задать имя категории в окне New Keyword и щёлкнуть по кнопке Add Category. Для присвоения свойства категории необходимо выбрать UDP из списка, затем категорию из списка категорий и щёлкнуть по кнопке Update. Одна категория может объединять несколько свойств, в то же время одно свойство может входить в несколько категорий. Свойство типа List может содержать массив предварительно определённых значений. Для определения области значений UDP типа List следует задать значение свойства в окне New Keyword и щёлкнуть по кнопке Add Member. Значения из списка можно редактировать.

Каждой работе можно поставить в соответствие набор UDP. Для этого следует щёлкнуть правой кнопкой мыши по работе и выбрать пункт меню UDP. В закладке UDP Values диалога IDEF0 Activity Properties можно задать значения UDP. Результат задания можно проанализировать в отчёте Diagram Object Report (меню Tools/Report/Diagram Object Report).

Контрольные вопросы

1 Объясните назначение UDP.

2 Какие действия необходимо выполнить для создания нового свойства UDP?

3 Как внести ключевые слова UDP?

4 Где отображается результат задания значений UDP?

5 Как назначить UDP какой-либо работе?

6 Как создать отчет по UDP и каковы его опции?

Лабораторная работа № 8. Создание модели TO-BE (реинжиниринг бизнес-процессов)

Цель работы: изучить создания модели TO-BE.

Задание к лабораторной работе.
Создать модель TO-BE для индивидуального задания.

Методические указания

Модель TO-BE создается на основе анализа модели AS-IS. Анализ может проводиться как по формальным признакам (отсутствие выходов или управлений у работ, отсутствие обратных связей и т. д.), так и по неформальным – на основе знаний предметной области.

Допустим, в результате анализа принимается решение реорганизовать функции производства и тестирования компьютеров и оставить функциональности «***Продажи и маркетинг***» и «***Отгрузка и получение***» пока без изменений.

Принято решение сформировать отдел дизайна, который должен формировать конфигурацию компьютеров, разрабатывать корпоративные стандарты, подбирать приемлемых поставщиков, разрабатывать инструкции по сборке, процедуры тестирования и устранения неполадок для всего производственного отдела.

Работа «***Сборка и тестирование компьютеров***» должна быть реорганизована и названа ***«Производство продукта»***. Будут созданы работы «***Разработать конфигурацию***», «***Планировать производство***» и «***Собрать продукт***».

Рассмотрим новые роли персонала. Дизайнер должен разрабатывать систему, стандарты на продукцию, документировать и передавать спецификации в отдел маркетинга и продаж. Он должен определять, какие компоненты (аппаратные и программные) должны закупаться для сборки компьютеров, обеспечивать документацией и управлять процедурами сборки, тестирования и устранения неполадок.

Функции диспетчера в работе «***Сборка и тестирование компьютеров***» должны быть заменены на функции планировщика.

Планировщик должен обрабатывать заказы клиентов и генерировать заказы на сборку, получить коммерческий прогноз из отдела маркетинга и формировать требования на закупку компонентов и собирать информацию от поставщиков.

Диспетчер должен составлять расписание производства на основании заказов на сборку, полученных в результате работы «***Планировать производство***», получать копии заказов клиентов и отвечать за упаковку и комплектацию заказанных компьютеров, передаваемых в работу «***Отгрузка и получение***».

Контрольные вопросы

1 Назначение видов модели AS-IS или TO-BE.
2 Что является основой для создания модели TO-BE?
3 В чем заключается анализ модели AS-IS по формальным признакам?
4 В чем заключается анализ модели AS-IS по неформальным признакам?

Лабораторная работа № 9. Создание диаграммы DFD

Цель работы: освоить методику создания диаграмм DFD.

Задание к лабораторной работе.
Создать диаграмму DFD для индивидуального задания.

Методические указания

Диаграммы потоков данных (Data Flow Diagramming) показывают, как каждый процесс преобразует входные данные в выходные, и выявляют отношения между процессами. DFD-диаграммы используются как дополнение к модели IDEF0 для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет моделируемую систему как сеть связанных работ. Основные компоненты DFD – процессы или работы, внешние сущности, потоки данных, накопители данных хранилища.

Для того чтобы дополнить модель IDEF0 диаграммой DFD, нужно в процессе декомпозиции в диалоге Activity Box Count выбрать радио-кнопку DFD. В палитре инструментов на диаграмме DFD появляются новые кнопки:

External Reference – добавить в диаграмму внешнюю ссылку; Data store – добавить в диаграмму хранилище данных;

Diagram Dictionary Editor – ссылка на другую страницу. В отличие от IDEF0 этот инструмент позволяет направить стрелку на любую диаграмму (а не только на верхний уровень).

В отличие от стрелок IDEF0, которые представляют собой жёсткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой. Это представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на физические характеристики системы – движение объектов, хранение объектов, поставка и распространение объектов.

Внешние сущности изображают входы в систему и выходы из неё. Они представляются в виде прямоугольника с тенью и располагаются по краям диаграммы. Одну внешнюю сущность можно использовать многократно на одной или нескольких диаграммах. Такой приём применяют, чтобы не рисовать слишком длинных и запутанных стрелок.

Потоки работ изображаются стрелками и описывают движение объектов из

одной части системы в другую. Поскольку в DFD каждая сторона работы не имеет чёткого назначения, 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 отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое. В материальных системах хранилища изображаются там, где объекты ожидают обработки, например в очереди. В системах обработки информации хранилища являются механизмом, который позволяет сохранить данные для последующих процессов.

В DFD стрелки могут сливаться и разветвляться, что позволяет описать их декомпозицию. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя.

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 имеет префикс D и уникальный номер, например D5. Каждая внешняя сущность имеет префикс E и уникальный номер, например E5.

Контрольные вопросы

1 Что описывает диаграмма DFD?
2 Перечислите составные части диаграммы DFD.
3 Объясните механизм дополнения диаграммы IDEF0 диаграммой DFD.
4 Что называется объектом-ссылкой?
5 Как добавить объект-ссылку?
6 В чем заключается правило DFD преобразования граничных стрелок.
7 Для чего используются двунаправленные стрелки?
8 Для чего применяется инструмент Off-Page Reference?
9 Для чего применяется межстраничная ссылка на диаграмме?
10 Объясните механизм дополнения диаграммы IDEF0 диаграммой DFD.

Лабораторная работа № 10. Построение диаграммы классов и схемы базы данных

Цель работы: научиться строить диаграммы классов с применением языка UML. Создать и описать логическую схему базы данных.

Задание к лабораторной работе.

Создать диаграмму классов для индивидуального задания.

Методические указания

Классы – это самые важные строительные блоки любой объектно–ориентированной системы. Они представляют собой описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. Графически класс изображается в виде прямоугольника. У каждого класса должно быть имя, отличающее его от других классов. Имя класса – это текстовая строка, содержащая различные символы. Имя может занимать одну или несколько строк. На практике для именования класса используют одно или несколько коротких существительных, взятых из словаря моделируемой системы. Обычно каждое слово в имени класса пишется с заглавной буквы, например: Customer (Клиент), TemperatureSensor (ДатчикТемпературы).

Для класса можно указать список атрибутов. Атрибут – это именованное свойство класса, включающее описание множества значений, которые могут принимать экземпляры этого свойства. Класс может иметь любое число атрибутов или не иметь их вовсе. Атрибут представляет некоторое свойство моделируемой сущности, общее для всех объектов данного класса. Например, при моделировании клиентов можно задавать фамилию, адрес, номер телефона и дату рождения. Атрибуты представляются в разделе, который расположен под именем класса.

Также для класса можно указать операции. Операция – это описание того, что позволено делать с объектом. У всех объектов класса имеется общий набор операций. Класс может содержать любое число операций или не содержать их вовсе. Например, для всех объектов класса Rectangle (Прямоугольник) из библиотеки для работы с окнами, содержащейся в пакете awt языка Java, определены операции перемещения, изменения размера и опроса значений свойств. Часто (хотя не всегда) обращение к операции объекта изменяет его состояние или его данные. Операции класса изображаются в разделе, расположенном под разделом с атрибутами класса.

Классы используются для составления словаря разрабатываемой информационно-аналитической системы. Это могут быть абстракции, являющиеся частью предметной области, либо классы, на которые опирается реализация. С их помощью описывают программные, аппаратные или чисто концептуальные сущности.

При построении модели информационно–аналитической системы оказы-

вается, что классы редко существуют автономно. Как правило, они разными способами взаимодействуют между собой. Это значит, что в ходе моделирования необходимо не только идентифицировать сущности, составляющие словарь системы, но и описать, как они соотносятся друг с другом.

Существует три вида отношений, особенно часто применяемых в диаграммах классов:

– зависимости, которые описывают существующие между классами отношения использования (включая отношения уточнения, трассировки и связывания). Графически отображаются пунктирными стрелками;

– обобщения, связывающие обобщённые классы со специализированными. Графически отображаются в виде стрелки с белым окончанием;

– ассоциации, представляющие структурные отношения между объектами. Графически отображаются как прямая линия.

Каждый из этих типов отношений позволяет по-разному комбинировать классы на диаграммах.

Диаграммой классов называют диаграмму, на которой показано множество классов, интерфейсов, кооперации и отношений между ними. Её изображают в виде множества вершин и дуг. Пример диаграммы классов представлен на рисунке 13.

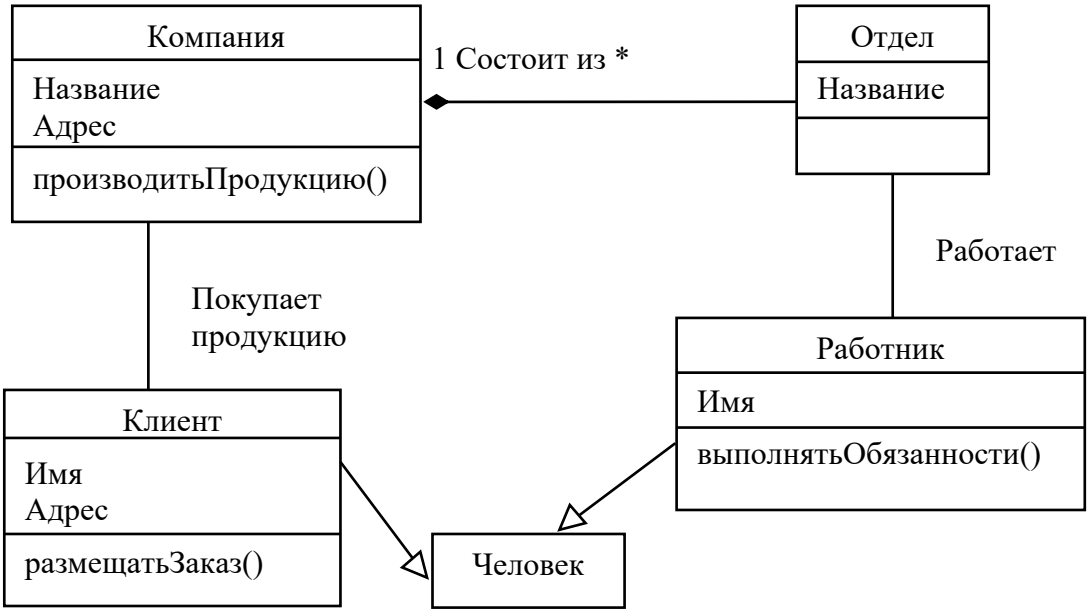


Рисунок 13 – Диаграмма классов

Диаграммы классов применяют для моделирования статического вида системы. Обычно диаграммы классов используются в следующих целях:

– для моделирования словаря системы. Оно предполагает принятие решения о том, какие абстракции являются частью системы, а какие – нет. С помощью диаграмм классов можно определить эти абстракции и их обязанности;

– для моделирования логической схемы базы данных. Логическую схему можно представить как чертёж концептуального проекта базы данных с

помощью диаграмм классов.

Многие моделируемые системы содержат устойчивые объекты, которые можно сохранять в базе данных и извлекать при необходимости. Для этого чаще всего используют реляционные, объектно-ориентированные или гибридные объектно-реляционные базы данных. UML позволяет моделировать логические схемы баз данных и сами физические базы, т. к. включает в себя как частный случай диаграммы «сущность-связь» (ER-диаграммы), которые часто используются для логического проектирования баз данных. Но если в классических ER-диаграммах внимание сосредоточено только на данных, то диаграммы классов позволяют моделировать поведение. В реальной базе данных эти логические операции трансформируются в триггеры или хранимые процедуры. Моделирование схемы производится следующим образом:

– определить классы модели, состояние которых должно сохраняться после завершения работы создавшего их приложения;

– создать содержащую эти классы диаграмму и описать их как устойчивые с помощью стандартного помеченного значения persistent;

– раскрыть структурные особенности классов. В общем случае надо детально описать их атрибуты и обратить внимание на ассоциации и их кратности;

– найти типичные структурные образцы, усложняющие проектирование физической базы данных, например циклические ассоциации, ассоциации «один к одному» и n-арные ассоциации. При необходимости создать промежуточные абстракции для упрощения логической структуры;

– рассмотреть поведение этих классов, раскрывая операции, важные для доступа к данным и поддержания их целостности;

– стараться использовать в работе инструментальные средства, позволяющие преобразовать логический проект в физический.

Содержание диаграмм классов без труда отображается на объектно-ориентированные языки программирования. Перевод диаграммы классов на программный язык производится следующим образом:

– определить правила отображения модели на один или несколько языков программирования по выбору;

– в зависимости от семантики выбранных языков придётся отказаться от использования некоторых возможностей UML;

– применять помеченные значения;

– пользоваться инструментальными средствами для отображения модели на объектно-ориентированные языки программирования.

Хорошо структурированная диаграмма классов:

– заостряет внимание только на одном аспекте статического вида системы;

– содержит лишь элементы, существенные для понимания данного аспекта;

– показывает детали, соответствующие требуемому уровню абстракции;

– не настолько лаконична, чтобы ввести в заблуждение.

Принципы построения диаграммы классов:

– давать диаграмме имя, связанное с её назначением;

– располагать элементы так, чтобы минимизировать число пересечений;

– пространственно организовывать элементы так, чтобы семантически близкие сущности располагались рядом;

– привлекать внимание к важным особенностям диаграммы, используя примечания и цвет;

– не показывать слишком много разных видов отношений; в диаграмме классов должны доминировать отношения какого–либо одного вида.

Контрольные вопросы

1 Назначение диаграммы классов.
2 Как определяется «Имя класса»?
3 Что такое «Операция класса»?
4 Общий формат записи «Атрибут класса».
5 Назовите возможные отношения между классами.
6 Перечислите графические элементы диаграммы классов.

Лабораторная работа № 11. Построение диаграммы деятельности

Цель работы: научиться строить диаграммы деятельности с применением языка UML.

Задание к лабораторной работе.

Создать диаграмму деятельности для индивидуального задания. Оформить и описать алгоритмы основных операций.

Методические указания

Диаграммы деятельности используются для моделирования динамических аспектов поведения системы. Как правило, они применяются, чтобы промоделировать последовательные (а иногда и параллельные) шаги вычислительного процесса. С помощью диаграмм деятельности можно также моделировать жизнь объекта, когда он переходит из одного состояния в другое в разных точках потока управления. Диаграммы деятельности могут использоваться самостоятельно для визуализации, специфицирования, конструирования и документирования динамики совокупности объектов, но они пригодны также и для моделирования потока управления при выполнении некоторой операции. Диаграмма деятельности – это своеобразная блок–схема, которая описывает последовательность выполнения операций во времени. Деятельность – это некоторый относительно продолжительный этап выполнения в автомате. В конечном итоге деятельность сводится к некоторому действию, которое составлено из атомарных вычислений, приводящих к изменению состояния системы или возврату значения. Действие может заключаться в вызове другой операции, посылке сигнала, создании или

уничтожении объекта либо в простом вычислении – скажем, значения выражения. Изображается действие в виде прямоугольника со скруглёнными углами.

Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход представляется простой линией со стрелкой. Поток управления должен где-то начинаться и заканчиваться (разумеется, если это не бесконечный поток, у которого есть начало, но нет конца). На диаграмме деятельности можно задать как начальное состояние (закрашенный кружок), так и конечное (закрашенный кружок внутри окружности).

Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования потока управления. Как и в блок-схеме, можно включить в модель ветвление, которое описывает различные пути выполнения в зависимости от значения некоторого булевского выражения. Графически точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить – два или более. Для каждого исходящего перехода задается булевское выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов эти сторожевые условия не должны одновременно принимать значение «истина», иначе поток управления окажется неоднозначным. Но эти условия должны покрывать все возможные варианты, иначе поток остановится.

Реализовать итерацию на диаграмме деятельности можно, если ввести два состояния действия – в первом устанавливается значение счетчика, во втором оно увеличивается – и точку ветвления, вычисление в которой показывает, следует ли прекратить итерации.

Простые и ветвящиеся последовательные переходы в диаграммах деятельности используются чаще всего. Однако можно встретить и параллельные потоки, и это особенно характерно для моделирования бизнес–процессов. В UML для обозначения разделения и слияния таких параллельных потоков выполнения используется синхронизационная черта, которая рисуется в виде жирной вертикальной или горизонтальной линии. Каждый из параллельно выполняющихся потоков управления существует в контексте независимого активного объекта, который, как правило, моделируется либо процессом, либо вычислительной нитью.

Должен поддерживаться баланс между точками разделения и слияния. Это означает, что число потоков, исходящих из точки разделения, должно быть равно числу потоков, приходящих в соответствующую ей точку слияния. Деятельности, выполняемые в параллельных потоках, могут обмениваться друг с другом информацией, посылая сигналы. Такой способ организации взаимодействия последовательных процессов называется сопрограммами.

При моделировании течения бизнес-процессов иногда бывает полезно разбить состояния деятельности на диаграммах деятельности на группы, каждая из которых представляет отдел компании, отвечающий за ту или иную работу. В UML такие группы называются дорожками, поскольку визуально каждая группа отделяется от соседних вертикальной чертой, как плавательные дорожки в бассейне.

Каждой присутствующей на диаграмме дорожке присваивается уникальное имя. Никакой глубокой семантики дорожка не несет, разве что может отражать некоторую сущность реального мира. Каждая дорожка представляет сферу ответственности за часть всей работы, изображённой на диаграмме, и в конечном счёте может быть реализована одним или несколькими классами. На диаграмме деятельности, разбитой на дорожки, каждая деятельность принадлежит ровно одной дорожке, но переходы могут пересекать границы дорожек.

Имеется некоторая связь между дорожками и параллельными потоками выполнения. Концептуально деятельность внутри каждой дорожки обычно – но не всегда – рассматривается отдельно от деятельности в соседних дорожках. Это разумно, поскольку в реальном мире подразделения организации, представленные дорожками, как правило, независимы и функционируют параллельно. Пример диаграммы деятельности представлен на рисунке 14.

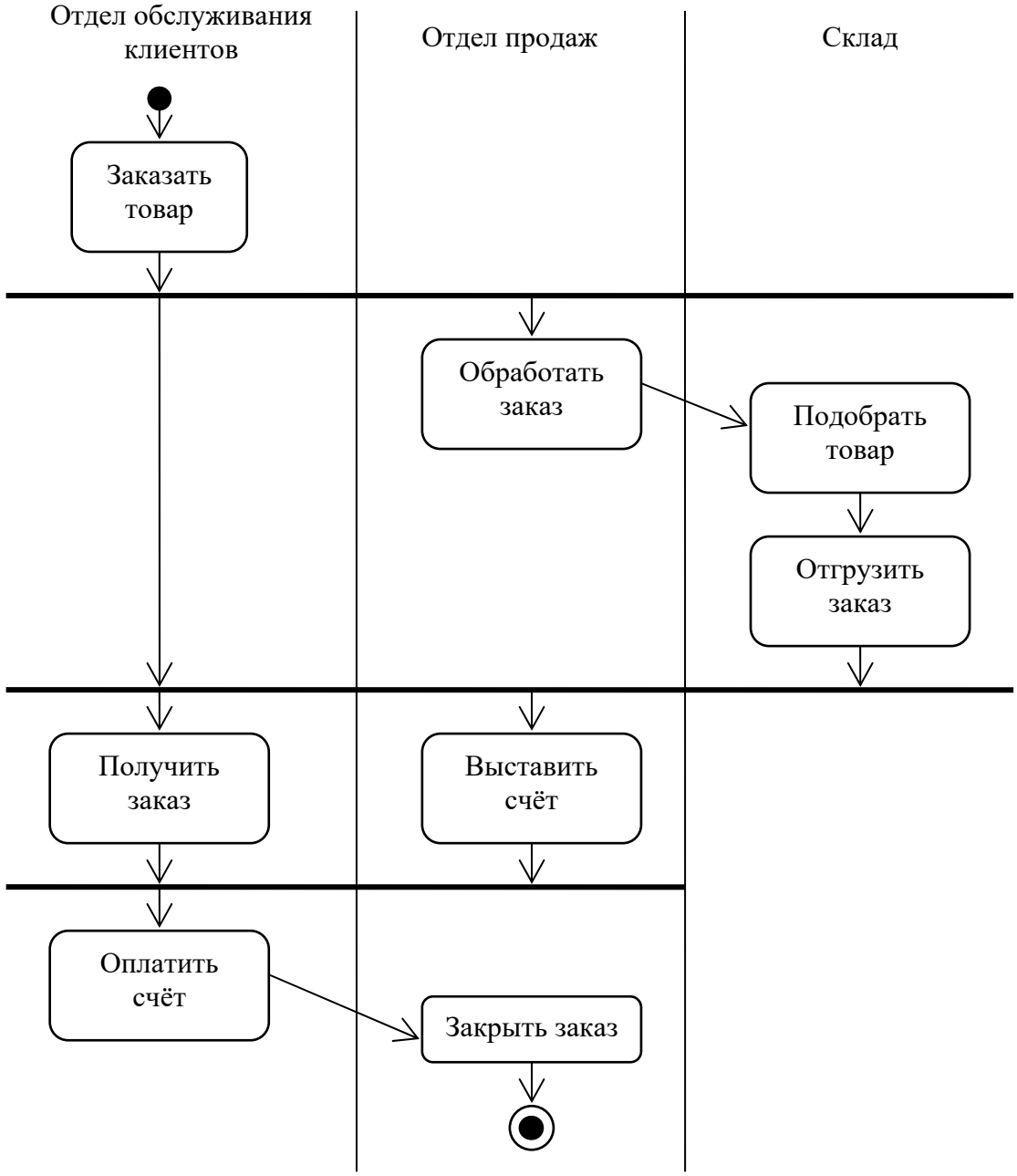


Рисунок 14 – Диаграмма деятельности

Использовать диаграмму деятельности для моделирования некоторого динамического аспекта системы можно в контексте любого элемента модели. Но чаще всего они рассматриваются в контексте системы в целом, подсистемы, операции или класса. Можно присоединять диаграммы деятельности к прецедентам и кооперациям. При моделировании динамических аспектов системы диаграммы деятельности применяются в основном двумя способами:

– для моделирования рабочего процесса. Здесь внимание фокусируется на деятельности с точки зрения актёров, которые сотрудничают с системой. Рабочие процессы часто оказываются с внешней, обращённой к пользователю стороны системы, и используются для описания бизнес–процессов разрабатываемой системы. При этом особенно важное значение имеет моделирование траекторий объектов;

– для моделирования операции. В этом случае диаграммы деятельности используются как блок-схемы для моделирования деталей вычислений. Для такого применения особенно важно моделирование точек ветвления, разделения и слияния. При этом контекст диаграммы деятельности включает параметры операции и её локальные объекты.

Программная система не существуют изолированно; всегда имеется некоторый контекст, в рамках которого она функционирует, причем он включает актёров, взаимодействующих с системой. Как правило, программное обеспечение работает в контексте бизнес-процессов более высокого уровня (рабочих процессов). Моделировать эти бизнес-процессы можно с помощью диаграмм деятельности. Для того чтобы построить модель рабочего процесса, следует:

– выделить какой-либо участок рабочего процесса. В сложных системах невозможно отразить все последовательности на одной диаграмме;

– выбрать бизнес-объекты, на которые возложена ответственность высокого уровня за части всего рабочего процесса. Это могут быть реальные сущности, вошедшие в системный словарь, или более абстрактные объекты. Следует создать отдельную дорожку для каждого бизнес-объекта;

– определить предусловия для начального состояния рабочего процесса и постусловия для его конечного состояния, чтобы задать границы процесса;

– начиная с исходного состояния описать деятельности и действия, выполняемые в различные моменты времени, а затем отразить их на диаграмме деятельности в виде состояний деятельности или действий;

– сложные действия или множества действий, встречающиеся многократно, следует свернуть в состояния деятельности и для каждого из таких состояний составить отдельную диаграмму деятельности;

– изобразить переходы, соединяющие состояния деятельностей и действий. Сначала нужно сосредоточиться на последовательных потоках, затем перейти к ветвлениям и затем рассмотреть разделения и слияния;

– если в рабочий процесс вовлечены важные объекты, изобразить их на диаграмме деятельности. Иногда следует показать изменение значений и состояний таких объектов, чтобы прояснить суть траектории каждого.

Диаграмму деятельности можно присоединить к любому элементу модели для описания поведения этого элемента. Чаще всего диаграммы деятельности присоединяются к операциям. В этом случае они становятся блок-схемой выполняемых действий. Основное преимущество диаграммы деятельности заключается в том, что все её элементы семантически связаны с лежащей в её основе богатой моделью. Моделирование операции состоит из следующих шагов:

– выявить абстракции, относящиеся к операции. Сюда относятся параметры операции (включая тип возвращаемого значения, если таковое имеется), атрибуты объемлющего класса и некоторых соседних классов;

– определить предусловия в начальном состоянии и постусловия в конечном состоянии операции;

– начиная с исходного состояния операции, описать деятельности и действия, протекающие во времени, и изобразить их на диаграмме деятельности в виде состояний деятельности или действий;

– при необходимости использовать точки ветвления для описания условных переходов и итераций;

– лишь в том случае, если владельцем операции является активный класс, использовать точки разделения и слияния для описания параллельных потоков выполнения, если есть необходимость.

Использование диаграмм деятельности в качестве блок-схем превращает UML в язык визуального программирования. Можно нарисовать блок-схему для каждой операции, но более естественно кодировать тело операции на некотором языке программирования. Использование диаграмм деятельности для моделирования операции становится разумным, когда эта операции сложна, так что разобраться в ней, глядя только на код, достаточно трудно.

При создании диаграмм деятельности следует помнить, что они моделируют срез некоторых динамических аспектов поведения системы. С помощью одной диаграммы деятельности невозможно охватить все динамические аспекты системы. Вместо этого следует использовать разные диаграммы деятельности для моделирования динамики рабочих процессов или отдельных операций.

Хорошо структурированная диаграмма деятельности:

– сконцентрирована на описании одного аспекта динамики системы;

– содержит только элементы, существенные для понимания этого аспекта;

– представляет лишь детали, соответствующие своему уровню абстракции;

– не настолько краткая, чтобы читатель упустил из виду важные аспекты.

Принципы построения диаграммы деятельности:

– давать диаграмме имя, соответствующее её назначению;

– начинать с моделирования главного потока. Ветвления, параллельность и траектории объектов являются второстепенными деталями, которые можно изобразить на отдельной диаграмме;

– располагать элементы так, чтобы число пересечений было минимальным;

– использовать примечания и цвет, чтобы привлечь внимание к важным особенностям диаграммы.

Контрольные вопросы

1 Дайте определение понятию «диаграмма деятельности».

2 Опишите назначение диаграммы деятельности.

3 Дайте определение понятиям «состояние деятельности» и «состояние действия». Графическое изображение состояния.

4 Приведите пример ветвления и параллельных потоков управления процессами на диаграмме деятельности.

5 Какие переходы используются на диаграмме деятельности?

6 Что представляет собой дорожка на диаграмме деятельности?

7 Как графически изображаются объекты на диаграмме деятельности?

Лабораторная работа № 12. Разработка пользовательского интерфейса

Цель работы: научиться разрабатывать простейший пользовательский интерфейс.

Задание к лабораторной работе.

В соответствии с вариантом задания, определяющим предметную область разработать пользовательский интерфейс.

Методические указания

Пользовательский интерфейс – это интерфейсное приложение, с которым пользователь взаимодействует для использования программного обеспечения. Пользователь может манипулировать программным и аппаратным обеспечением и управлять им с помощью пользовательского интерфейса. Основа взаимодействия – диалоги.

Диалог – регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи: обмен информацией и координация действий. Каждый диалог состоит из отдельных процессов ввода-вывода, которые физически обеспечивают связь пользователя и компьютера.

Пользовательский интерфейс является частью программного обеспечения и спроектирован таким образом, чтобы обеспечить понимание пользователем программного обеспечения. Пользовательский интерфейс обеспечивает фундаментальную платформу для взаимодействия человека с компьютером.

Разработка пользовательского интерфейса включает те же основные этапы, что и разработка программного обеспечения:

– постановка задачи – определение типа интерфейса и общих требований к нему;

– анализ требований и определение спецификаций – определение сценариев

использования и пользовательской модели интерфейса;

– проектирование – проектирование диалогов и их реализация в виде процессов ввода-вывода;

– реализация – программирование и тестирование интерфейсных процессов.

Для получения эффективного результата разработки ПИ интерфейса используют различные подходы к проектированию.

1 *Подход, ориентированный на пользователя,* основным содержанием этого подхода является ориентация на пользователя, т. е. в первую очередь необходимо узнать, что хочет пользователь получить от проектируемого интерфейса. Далее в процессе проектирования полученные требования реализуются в продукте. При сборе информации используются методы наблюдения за работой пользователя, проводятся интервью.

2 *Системный подход.* Пользователь рассматривается как маленькая интеллектуальная часть системы «человек - программный продукт».

3 *Деятельностный подход.* Изучается деятельность пользователя в целом, и постепенно оптимизируются её отдельные моменты.

4 *Итеративный подход* – метод последовательных приближений. Суть итеративного подхода заключается в создании изначально самого простейшего прототипа с целью показать заказчику и затем постепенно дорабатывать прототип, основываясь на реакции заказчика после каждого шага доработки.

5 *Экспертный подход.* Заключается в следующем: эксперт собирает важную, по его мнению, информацию, ведёт переговоры с заказчиком, задаёт нужные вопросы. На основе полученной информации создаётся интерфейс.

6 *Целеориентированный подход* проектирования. Разработка интерфейса ориентируется на цель, которая будет достигаться данным программным продуктом.

7 *Средоориентированный подход.* Разрабатывается среда интерфейса как место деятельности оператора.

При разработке интерфейса целесообразно гибко пользоваться указанными подходами, учитывая при выборе методов: назначение разрабатываемого продукта, целевую аудиторию, время и бюджет разработки.

Контрольные вопросы

1 Пользовательский интерфейс.
2 Этапы разработки пользовательского интерфейса.
3 Подходы к проектированию пользовательского интерфейса.

Лабораторная работа № 13. Разработка информационно-аналитической системы

Цель работы: знакомство со стандартами, регламентирующими жизненный цикл разработки информационных систем. Приобретение практических навыков при разработке программного документа Техническое задание.

Задание к лабораторной работе.

В соответствии с вариантом задания, определяющим предметную область, разработать документ Техническое задание на создание информационно–аналитической системы.

Методические указания

Для обеспечения потребностей по разработке программного обеспечения (ПО) ИС необходимо взаимосвязанное совершенствование технических решений, технологий проектирования и программирования, инструментальных средств, а также обучения специалистов.

Стандарты для процессов, инструментальных средств и данных, которые отражают лучшую практику и защищают от неблагоприятных последствий, играют определенную роль в обеспечении указанных потребностей, в частности, поддерживая доверие потребителя к продуктам, услугам и технологиям разработки ПО.

Использование стандартов при разработке ПО ИС позволит обеспечить:

– повышение надежности;

– повышение эффективности применения и снижение затрат в сфере сопровождения программных средств (ПС);

– увеличение коэффициента повторного использования ПС общего назначения;

– повышение объективности оценок качества ПС;

– создание условий для конкуренции разработчиков на внутреннем рынке ПС;

– обеспечение конкурентоспособности отечественных ПС на мировом рынке.

Стандарты на создание и развитие автоматизированных систем (АС) – обобщенные, но воспринимаемые как весьма жесткие по структуре жизненного цикла (ЖЦ) и проектной документации. Наиболее популярными можно считать ГОСТ 34.601–90 *Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания* и ГОСТ 34.602–89 *Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы*. Введение единой, достаточно качественно определенной терминологии, наличие достаточно разумной классификации работ, документов, видов обеспечения и др. способствует более полной и качественной стыковке разных систем, что особенно важно в условиях, когда разрабатывается все больше сложных комплексных АС.

В последние годы в стране идет интенсивная работа по гармонизации государственных стандартов Республики Беларусь с международными стандартами ISO в области создания нормативной базы управления жизненным циклом программных средств и информационных систем. В основе стандартизации – СТБ ИСО/МЭК 12207–2003*«Информационные технологии. Процессы жизненного цикла программных средств*, который является аутентичным переводом международного стандарта ISO/IEC 12207:2008.

Техническое задание (ТЗ) на АС – утвержденный в установленном порядке документ, определяющий цели, требования и основные исходные данные, необходимые для разработки АС, и содержащий предварительную оценку экономической эффективности.

ТЗ содержит основные технические требования, предъявляемые к изделию, и исходные данные для разработки; в ТЗ указываются назначение объекта, область его применения, стадии разработки документации, её состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов, полученных в ходе предпроектного обследования.

Как инструмент коммуникации в связке общения заказчик-исполнитель, техническое задание позволяет:

– обеим сторонам:

а) представить готовый продукт;

б) выполнить проверку готового продукта по пунктам (приёмочное тестирование – проведение испытаний);

в) уменьшить число ошибок, связанных с изменением требований в результате их неполноты или ошибочности (на всех стадиях и этапах создания, за исключением испытаний);

– заказчику:

а) осознать, что именно ему нужно, опираясь на существующие на данный момент технические возможности и свои ресурсы;

б) требовать от исполнителя соответствия продукта всем условиям, оговорённым в ТЗ;

– исполнителю:

а) правильно понять суть задачи, показать заказчику «технический облик» будущего программного изделия или АС;

б) спланировать выполнение проекта и работать по намеченному плану;

в) отказаться от выполнения работ, не указанных в ТЗ.

Контрольные вопросы

1 Этапы развития проекта ИС.
2 Понятие модели жизненного цикла ПО ИС.
3 Обзор моделей жизненного цикла, их недостатки и преимущества.
4 Стадии и этапы создания АС в соответствии с ГОСТ 34.601–90.
5 Назначение, содержание стандарта ГОСТ 34.602–89.

6 Назначение, содержание и степень адаптивности стандарта СТБ ИСО/МЭК 12207–2003.

7 Назначение программного документа Техническое задание на создание АС. Порядок разработки, согласования и утверждения документа.

8 Состав и содержание документа ТЗ.

Лабораторная работа № 14. Отладка и тестирование информационно-аналитической системы

Цель работы: изучить общие понятия и принципы тестирования ИАС.

Задание к лабораторной работе.

В соответствии с вариантом задания, определяющим предметную область, составить тест создаваемой ИАС.

Методические указания

Тестирование – процесс выполнения программы с целью обнаружения ошибок. Шаги процесса задаются тестами.

Каждый тест определяет:

- свой набор исходных данных и условий для запуска программы;
- набор ожидаемых результатов работы программы.

Другое название теста – тестовый вариант. Полную проверку программы гарантирует *исчерпывающее тестирование.* Оно требует проверить все наборы исходных данных, все варианты их обработки и включает большое количество тестовых вариантов. Исчерпывающее тестирование во многих случаях невозможно – срабатывают ресурсные ограничения (прежде всего, ограничения по времени).

Целью проектирования тестовых вариантов является систематическое обнаружение различных классов ошибок при минимальных затратах времени и стоимости.

Тестирование обеспечивает:

- обнаружение ошибок;
- демонстрацию соответствия функций программы ее назначению;
- демонстрацию реализации требований к характеристикам программы;
- отображение надежности как индикатора качества программы.

Тестирование не может показать отсутствия дефектов (оно может выявить только присутствие дефектов).

Существуют два принципа тестирования программы:

1) функциональное тестирование (тестирование «черного ящика»);
2) структурное тестирование (тестирование «белого ящика»).

Тестирование «черного ящика».

Известны: функции программы.

Исследуется: работа каждой функции на всей области определения.

Эти тесты демонстрируют:

- как выполняются функции программ;
- как принимаются исходные данные;
- как вырабатываются результаты;
- как сохраняется целостность внешней информации.

При тестировании «черного ящика» рассматриваются системные характеристики программ, игнорируется их внутренняя логическая структура. Исчерпывающее тестирование, как правило, невозможно. Например, если в программе 10 входных величин и каждая принимает по 10 значений, то потребуется 10^{10} тестовых вариантов. Отметим также, что тестирование «черного ящика» не реагирует на многие особенности программных ошибок.

Тестирование «белого ящика».

Известна: внутренняя структура программы.

Исследуются: внутренние элементы программы и связи между ними.

Объектом тестирования здесь является не внешнее, а внутреннее поведение программы. Проверяется корректность построения всех элементов программы и правильность их взаимодействия друг с другом. Обычно анализируются управляющие связи элементов, реже – информационные связи. Тестирование по принципу «белого ящика» характеризуется степенью, в какой тесты выполняют или покрывают логику (исходный текст) программы.

Контрольные вопросы

1 Понятия и принципы тестирования.
2 Структурное тестирование (тестирование «белого ящика»).
3 Функциональное тестирование (тестирование «черного ящика»).

Список литературы

1 **Гвоздева, Т. В.** Проектирование информационных систем: технология автоматизированного проектирования. Лабораторный практикум: учебно-справочное пособие / Т. В. Гвоздева, Б. А. Баллод. – Санкт-Петербург; Москва; Краснодар: Лань, 2018. – 154 с.

2 **Голицына, О. Л.** Информационные системы: учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 2-е изд. – Москва: ФОРУМ; ИНФРА-М, 2016. – 416 с.

3 **Емельянова, Н. З.** Проектирование информационных систем: учебное пособие / Н. З. Емельянова, И. И. Папов, Т. Л. Партыко – Москва: Форум; ИНФРА-М, 2017. – 432 с.

4 **Маран, М. М.** Программная инженерия: учебное пособие / М. М. Маран. – Санкт-Петербург; Москва; Краснодар: Лань, 2018. – 196 с.: ил.

5 **Рыбальченко, М. В.** Архитектура информационных систем: учебное пособие для вузов / М. В. Рыбальченко. – Москва: Юрайт, 2016. – 91 с.