

ЛАБОРАТОРНАЯ РАБОТА №1

«СТРУКТУРНЫЙ АНАЛИЗ СИСТЕМЫ. РАЗРАБОТКА ДИАГРАММЫ ПОТОКОВ ДАННЫХ И СОСТАВЛЕНИЕ СПЕЦИФИКАЦИИ ПРОЦЕССОВ»

1.1 Цель работы

Целью данной работы является изучение методологии графического структурного анализа, построение структурной модели на основе диаграмм потоков данных и уточнение данной модели посредством спецификации процессов.

1.2 Задание на лабораторную работу

Разработать структурную модель информационной системы или её функционально законченной части.

1. Построить контекстную диаграмму потоков данных;
2. Осуществить декомпозицию контекстной диаграммы (построить диаграмму потоков данных 1-го уровня);
3. Осуществить декомпозицию диаграммы 1-го уровня (построить диаграмму потоков данных 2-го уровня);
4. Составить спецификации всех процессов диаграммы потоков данных 1-го уровня с учетом подпроцессов при декомпозиции.

При составлении спецификаций все процессы должны быть разбиты на равные по количеству процессов группы.

Лабораторная работа будет оцениваться в соответствии со следующими критериями:

	Удовл.	Хорошо	Отлично
Количество процессов на диаграмме потоков данных 1-го уровня	10	12	15
Диаграмма потоков данных 2-го уровня должна представлять собой декомпозицию любых N процессов диаграммы 1-го уровня	$N = 3$	$N = 5$	$N = 7$
Количество способов представления спецификации процессов (структурированный естественный язык, псевдокод, блок-схема) ¹	1	2	3

¹ Выбор способа представления спецификации процессов остается за студентом, кроме задания на «отлично», в данном случае необходимо представить спецификацию процессов всеми тремя предложенными способами.

1.3 Порядок выполнения работы

1. Получить вариант задания (тему) у преподавателя;
2. Изучить теоретический материал, изложенный в подразделе 1.4;
3. Разработать структурную модель информационной системы и составить спецификацию всех процессов DFD 1-го уровня
4. Ознакомиться с требованиями по содержанию отчета в подразделе 1.5;
5. Написать отчет о работе.

Для выполнения лабораторных работы можно воспользоваться любой средой моделирования или CASE-средством, которое поддерживает соответствующие структурные нотации диаграмм. Рекомендуется использовать бесплатное для некоммерческого использования CASE-средство [Software Ideas Modeler](#).

1.4 Теоретический материал

Диаграммы потоков данных (Data Flow Diagrams, DFD) — методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки и хранилища данных, к которым осуществляется доступ.

При использовании структурной модели для анализа и проектирования систему представляют в виде иерархии диаграмм потоков данных. На каждом следующем уровне иерархий происходит декомпозиция, то есть разбиение на структурные составляющие процессов, которые преобразуют входную информацию с заданным алгоритмом в выходную. Когда достигается требуемая глубина декомпозиции, процессы нижнего уровня сопровождаются спецификацией.

В основе структурной модели лежат понятия внешней сущности, процесса, хранилища (накопителя) данных и потока данных.

Внешняя сущность — материальный объект или физическое лицо, выступающие в качестве источников или приемников информации, например, заказчик, персонал, пользователь (тот, кто использует продукт), клиент (тот, кто приобретает продукт), склад, дата-центр, etc. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что объект или система находятся за пределами границ анализируемой информационной системы. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь контура (диаграммы) анализируемой системы, если это необходимо, или, наоборот, часть процессов системы может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

Процесс — преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и составление отчетов, программа, аппаратно-реализованное логическое устройство, скрипт для реализации функций взаимодействия пользователя с веб-сайтом, etc. Таким образом, физически преобразование может осуществляться программными средствами, вручную или специальными устройствами. Каждый процесс в системе имеет свой номер и связан с исполнителем, который осуществляет данное преобразование.

На верхних уровнях иерархии, когда процессы ещё не определены, вместо понятия «процесс» используют понятия «система» и «подсистема», которые обозначают соответственно систему в целом или её функционально законченную часть.

Накопитель данных представляет собой абстрактное устройство для хранения информации. Тип устройства и способы помещения, извлечения и хранения для такого устройства не детализируют. Физически это может быть база данных, файл, таблица в оперативной памяти, картотека на бумаге, etc.

Идентификатор накопителя данных, который может быть произвольным числом, предваряет буква «D». Имя накопителя выбирается из соображения наибольшей информативности при проектировании информационной системы.

Накопитель данных в общем случае является прообразом будущей базы данных. Описание самих данных, хранение которых обеспечивается в накопителе, должно быть отражено в семантической модели в виде диаграммы сущность-связь.

Поток данных — процесс передачи некоторой информации от источника к приемнику. Физически процесс передачи информации может происходить по кабелям под управлением программы или программной системы или вручную при участии устройств или людей вне проектируемой системы.




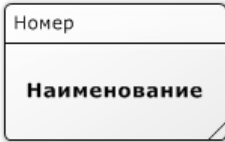


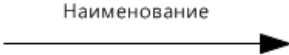
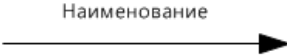
Таким образом, диаграмма иллюстрирует как потоки данных, порожденные некоторыми внешними сущностями, преобразуются соответствующими процессами (или подсистемами), сохраняются накопителями данных и передаются другим внешним сущностям — приемникам информации. В результате мы получаем структурную модель хранения (обработки) информации.

Построение иерархии диаграмм потоков данных начинают с диаграммы особого вида — контекстной диаграммы, которая определяет наиболее общий вид системы. На такой диаграмме показывают, как разрабатываемая система будет взаимодействовать с приемниками и источниками информации без указания исполнителей (процессов), иными словами описывают интерфейс между системой и внешним миром.

Если проектируемая система содержит большое количество внешних сущностей (более 10-ти), имеет распределенную природу или включает уже существующие подсистемы, то строят иерархии контекстных диаграмм. Иными словами структурную модель системы допустимо разбивать на несколько диаграмм потоков данных 1-го уровня.

Для графического описания диаграмм потоков данных в равной степени используют две нотации — Йордана (Yourdon) и Гейна-Сарсона (Gane-Sarson), которые отличаются синтаксисом. В настоящих методических указаниях используется нотация Гейна-Сарсона, однако при выполнении лабораторной работы студент вправе выбрать любую из представленных (табл.1.4).

Таблица 1.4 — Элементы диаграммы потоков данных

Понятие	Нотация Йордана	Нотация Гейна-Сарсона
Внешняя сущность		
Система, подсистема или процесс		
Накопитель данных		
Поток данных		

Над линией потока, направление которого обозначают стрелкой, указывают, какая конкретно информация в данном случае передается (рис. 1.4).

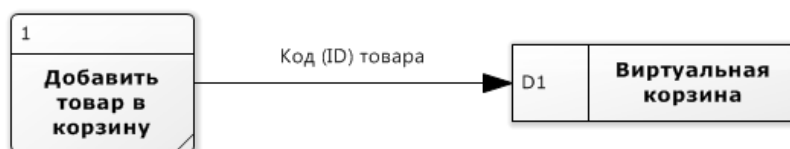


Рисунок 1.4 — Пример потока данных
(спецификация Гейна-Сарсона)

В нотации Гейна-Сарсона у процессов иногда добавляют дополнительное поле после наименования процесса — поле физической реализации, в котором указывают, какое подразделение организации, аппаратное устройство, программа или скрипт выполняют данный процесс (исполнитель). Данное поле не является обязательным для заполнения и во многих CASE-средствах в нотации Гейна-Сарсона оно отсутствует.

Спецификации процессов представляют собой описания алгоритмов задач, выполняемых процессами. Они содержат номер и/или имя процесса и описание логики (функции) данного процесса с указанием входных и выходных данных. Спецификацию процесса обычно представляют в виде структурированного естественного языка, псевдокодов, схем алгоритмов (например, блок-схемы), деревьев или таблиц решений, Flow-форм или диаграмм Насси-Шнейдермана etc.

Структурированный естественный язык представляет собой комбинацию формализма языка программирования и доступности естественного языка, включает в себя подмножество ключевых слов, организованных в определенные логические структуры. Ключевые слова в структурированном естественном языке выделяются заглавными буквами, а логика процессов выражается в виде комбинации последовательных конструкций выполнения команд, выбора и итераций.

1. Конструкции выполнения

ВЫПОЛНИТЬ «операция» // двойной слеш может служить для указания комментария

«другая операция»

Для разделения операций используется новая строка, однако можно задать и другие разделители, например точку с запятой. Возможен и такой вариант:

ВЫПОЛНИТЬ «операция»

ВЫПОЛНИТЬ «другая операция»

Но он более избыточен, чем предыдущий. Операции не обязательно заключать в кавычки, в настоящих методических указаниях это сделано для визуального комфорта.

2. Конструкции выбора

ЕСЛИ «условие» ТО

ВЫПОЛНИТЬ «операция»

ИНАЧЕ

ВЫПОЛНИТЬ «другая операция»

КОНЕЦ ЕСЛИ

В конструкциях выбора, как и во всем структурированном естественном языке допускается изменять нотацию написания, например, можно записать часть ключевых слов в одну строку. Кроме того, допускается выделять отступами вложенные конструкции для большей наглядности.

ИНАЧЕ ВЫПОЛНИТЬ «другая операция» КОНЕЦ ЕСЛИ

3. Конструкции итераций

ПОКА «условие»

ВЫПОЛНИТЬ «операция»

КОНЕЦ ПОКА

Набор ключевых слов также может быть дополнен в зависимости от структуры описываемой схемы алгоритма.

Пример спецификации процесса «Выполнение лабораторных работ» на структурированном естественном языке:

ВХОДНЫЕ ДАННЫЕ: событие «сессия» (true / false), важные дела (true / false)

ВЫХОДНЫЕ ДАННЫЕ: рейтинг за практические занятия

ЕСЛИ «сессия»

ВЫПОЛНИТЬ «переход в режим супергерой Человек-студент»,

«сделать все лабораторные работы за 24 часа»,

«получить минимальный рейтинг за практику (и осуждающий взгляд преподавателя)»,

«сократить продолжительность своей жизни за счет стресса от режима супергероя»

ИНАЧЕ

ЕСЛИ «есть более важные дела (чем выполнение ЛР)»

ВЫПОЛНИТЬ «просматривать картинки в социальных сетях; отвечать в

Интернете тем, кто не прав; ругать тиммейтов за то, что у них физиологические дефекты (руки не из того места, рак мозга в терминальной стадии, отсутствие признаков эволюции со времен кайнозойской эры и многие другие

определяющие характеристики); etc.»

ИНАЧЕ ВЫПОЛНИТЬ «сделать лабораторные работы с соблюдением сроков»,

«получить максимальный рейтинг за практику (и хорошее настроение)»

КОНЕЦ ЕСЛИ

КОНЕЦ ЕСЛИ

Псевдокод представляет собой компактный язык описания алгоритмов, использующий ключевые слова языков программирования, но опускающий несущественные подробности и специфический синтаксис. Псевдокод обычно опускает детали, несущественные для понимания алгоритма человеком. Такими несущественными деталями могут быть описания переменных, системно-зависимый код и подпрограммы. Таким образом, главная цель использования псевдокода — обеспечить понимание алгоритма человеком и сделать описание более воспринимаемым, чем исходный код на языке программирования. Синтаксис псевдокода может быть определен привязкой к какому-либо языку программирования, например, Pascal, C, PHP etc. В псевдокоде математические выражения обычно представляются в том виде, как их принято записывать в математике, а не в языках программирования, и некоторые фрагменты псевдокода могут быть фразами естественного языка (русского, английского etc.).

Пример спецификации процесса «Выполнение лабораторных работ» на псевдокоде:

ВХОДНЫЕ ДАННЫЕ: событие «сессия» (true / false), важные дела (true / false)

ВЫХОДНЫЕ ДАННЫЕ: рейтинг за практические занятия

```
if(exam) {  
    set superman_student;  
    do lab_work; get lose_lifetime;  
    get min_rating;  
} else {  
    if(important_task) {  
        do task;
```

```

    } else {
        do lab_work;
        get max_rating;
    }
}

```

Псевдокод можно представить и в нотации русского языка:

```

если (сессия) {
    супергерой_человек_студент = true;
    выполнить лаб_работа;
    продолжительность_жизни = продолжительность_жизни - RAND();
    // RAND() — функция псевдослучайных чисел
    получить мин_рейтинг;
} иначе {
    если(важная_задача) {
        выполнить важная_задача;
    } иначе {
        выполнить лаб_работа;
        получить макс_рейтинг;
    }
}

```

Базовые управляющие структуры псевдокода аналогичны соответствующим конструкциям в структурированном естественном языке с тем лишь отличием, что их синтаксис определяется за счет привязки к конкретному языку программирования.

В примере выполнения работы спецификация процессов на псевдокоде выполнена с привязкой к языку PHP с комментариями на русском языке и приведением упрощенной формы запросов к БД на языке SQL. Детализация и сложность описания процесса (вне зависимости от формы представления, будь то псевдокод или дерево решений) определяется исходя из соображений доступности (понятности) тому, для кого она предназначена. Например, процесс 2 «Регистрация» можно представить в следующей форме псевдокода:

```

if(существуют данные форм регистрации) {
    if(данные корректны) {
        result = запрос к БД на совпадение с указанными логином и паролем;
        // если совпадения не найдены, вернет false, в противном случае true
        if(result) {
            сообщение('Данные логин или почта уже заняты');
        } else { // если данные уникальны
            произвести запрос к БД на добавление нового пользователя;
            // и задать статус «ожидает подтверждения регистрации»
            отправить письмо пользователю для подтверждения регистрации;
            сообщение('Для подтверждения регистрации перейдите по ссылке в письме');
        }
    } else { сообщение('Проверьте введенные данные'); }
}

```

```

if(пользователь перешел по ссылке в письме) {
    обновить данные в БД; // изменить статус пользователя на «зарегистрирован»
    сообщение('Ваша учетная запись активирована');
}

```

Блок-схемы можно рассматривать как графическую альтернативу псевдокоду. В настоящих методических указаниях к работе подробно не рассматриваются элементы блок-схемы, так как подразумевается, что данный материал является базовым и был изучен ранее.

Пример спецификации процесса «Выполнение лабораторных работ» в виде блок-схемы:

ВХОДНЫЕ ДАННЫЕ: событие «сессия» (true / false), важные дела (true / false)
 ВЫХОДНЫЕ ДАННЫЕ: рейтинг за практические занятия

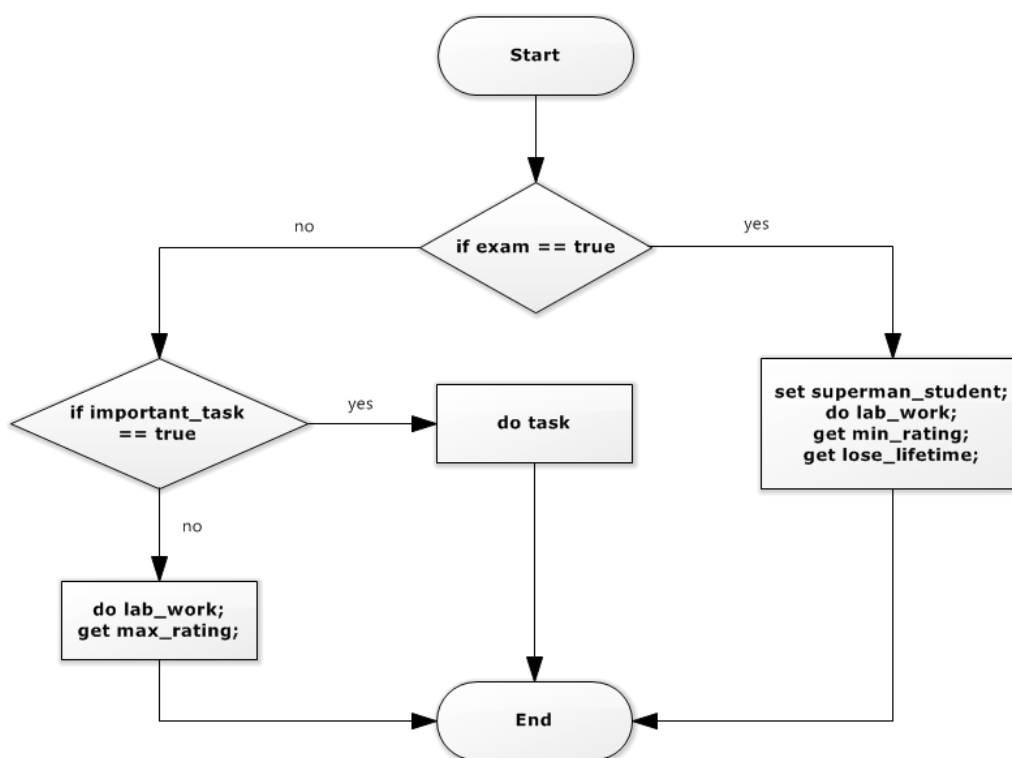


Рисунок 1.5 — Блок-схема процесса «Выполнение лабораторных работ»

Каждый из способов представления спецификации имеет свое определенное предназначение, которое зависит от пользователя данной спецификации (программист, аналитик, специалист по тестированию, etc.), его квалификации, типов решаемых задач, специфики конкретных процессов и самой информационной системы.

Во время выполнения работы при выборе, к какой группе способа представления спецификации отнести конкретный процесс (для оценок «хорошо» и «отлично»), следует руководствоваться соображениями удобства и простоты реализации.

1.5 Содержание отчета

Отчет о работе должен содержать:

1. Титульный лист
2. Цель работы
3. Задание на лабораторную работу и вариант
4. Структурная модель информационной системы «Наименование»
 - 4.1 Контекстная диаграмма
 - 4.2 Диаграмма потоков данных 1-го уровня
 - 4.3 Диаграмма потоков данных 2-го уровня
5. Спецификация процессов структурной модели
6. Выводы по работе
7. Используемые источники (4-6 источника)

1.6 Пример выполнения работы

Структурная модель информационной системы «Интернет-магазин»

Контекстная диаграмма потоков данных

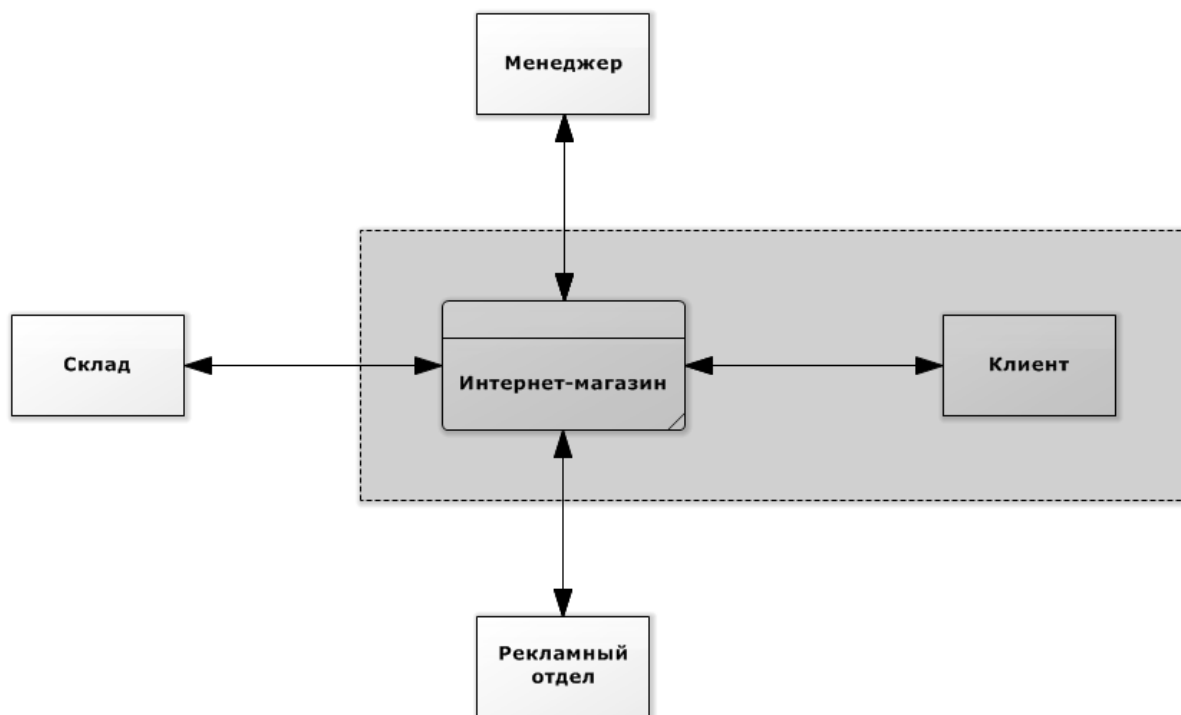


Рисунок 1.6.1 — Контекстная диаграмма системы «Интернет-магазин»

На рис. 1.6.1 серым контуром обозначен фрагмент контекстной диаграммы потоков данных, декомпозиция которого будет осуществлена. На данной контекстной диаграмме не указаны наименования потоков данных, это сделано для демонстрации общих отношений между системой и сущностями, но в целом для большей информативности контекстной диаграммы желательно выделить наименования основных потоков данных, как это сделано на рис. 1.6.2, кроме того, такой вид диаграммы используется при составлении спецификации программных требований, однако первый вариант диаграммы также допустим при работе.

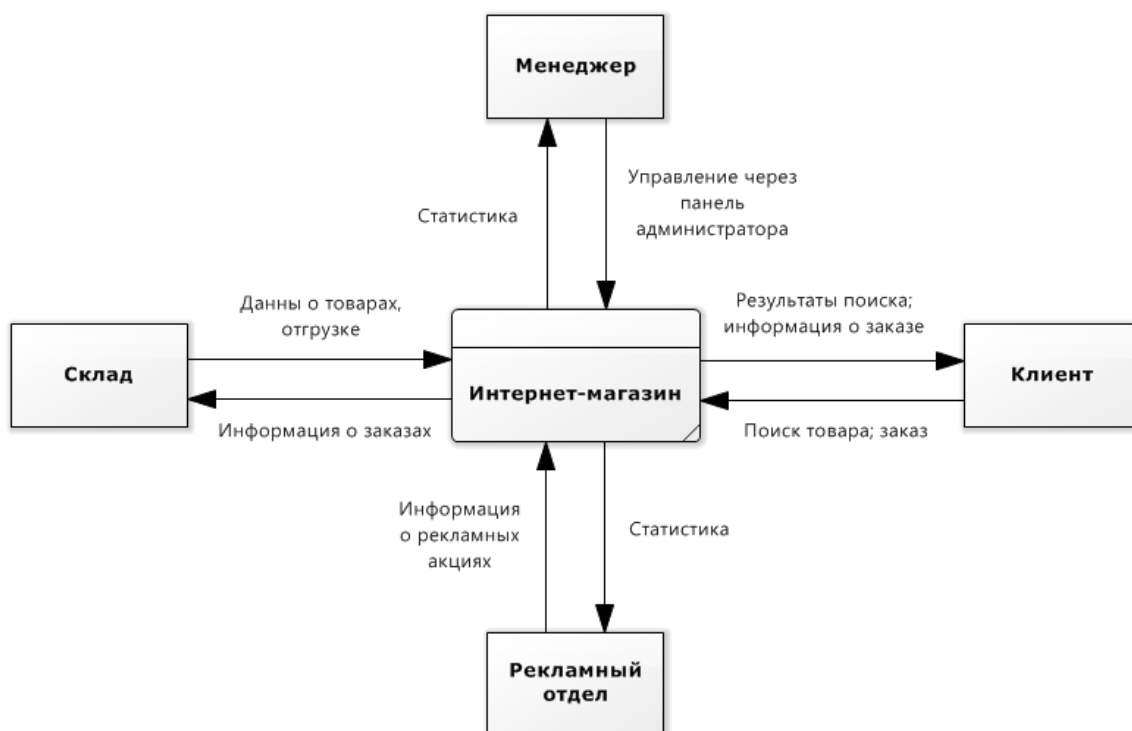


Рисунок 1.6.2 — Контекстная диаграмма системы «Интернет-магазин» с указанием наименований основных потоков данных

Если на контекстной диаграмме фигурирует одна система, то номер системы можно не указывать.

Диаграмма потоков данных 1-го уровня

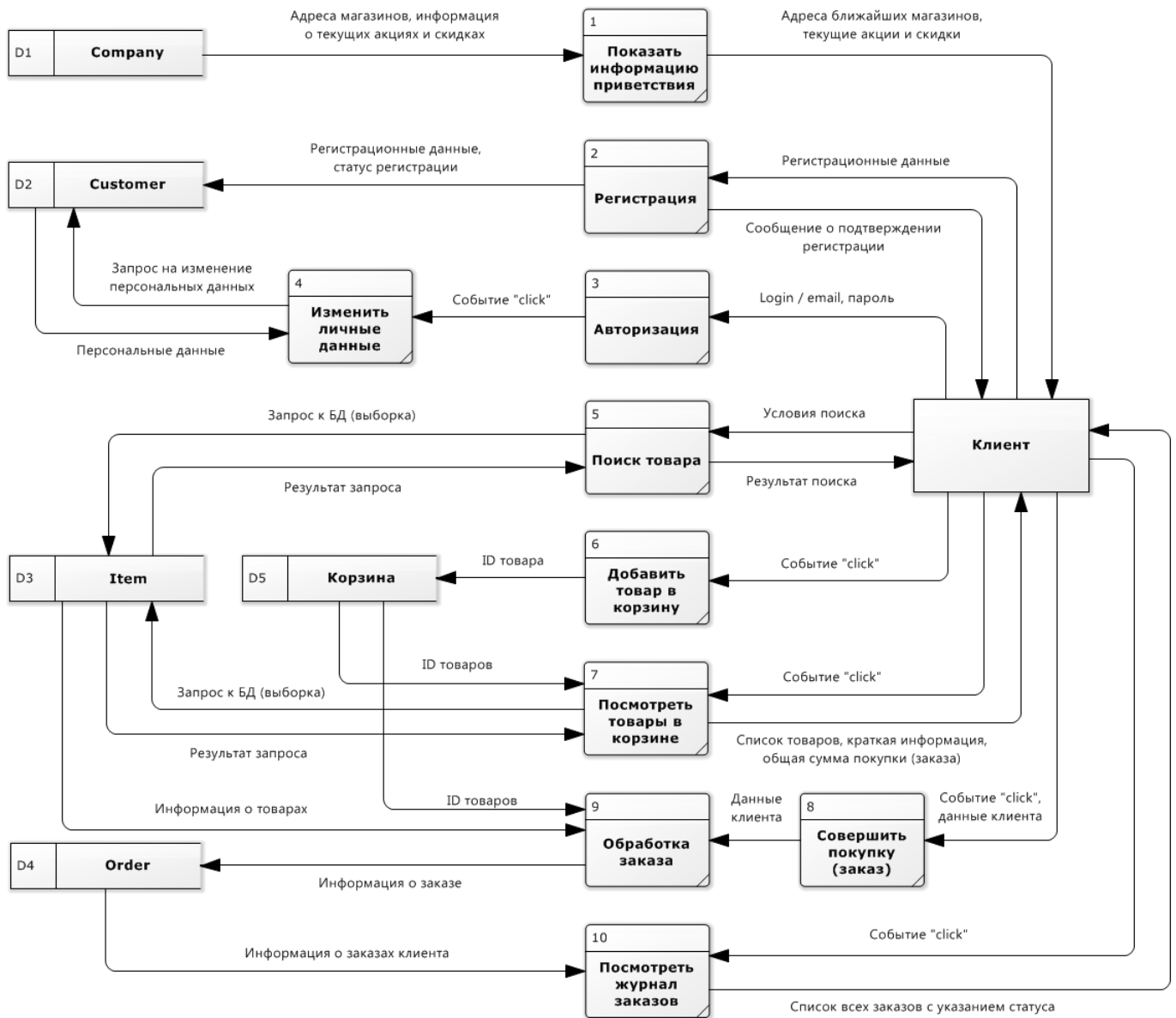


Рисунок 1.6.3 — Диаграмма потоков данных 1-го уровня для информационной системы «Интернет-магазин»

В Software Ideas Modeler:

1. Изменение типа линии:левой кнопкой мыши (ЛКМ) на поток → Line Style → Rectangular;
2. Изменение направления потока: ЛКМ на поток: Relationship → Reverse Relationship;
3. Экспорт диаграммы в изображение: Diagram → Export → Image.

Приведенную диаграмму потоков данных 1-го уровня (рис. 1.6.3) можно семантически структурировать, то есть разбить на ряд диаграмм потоков данных 1-го уровня:

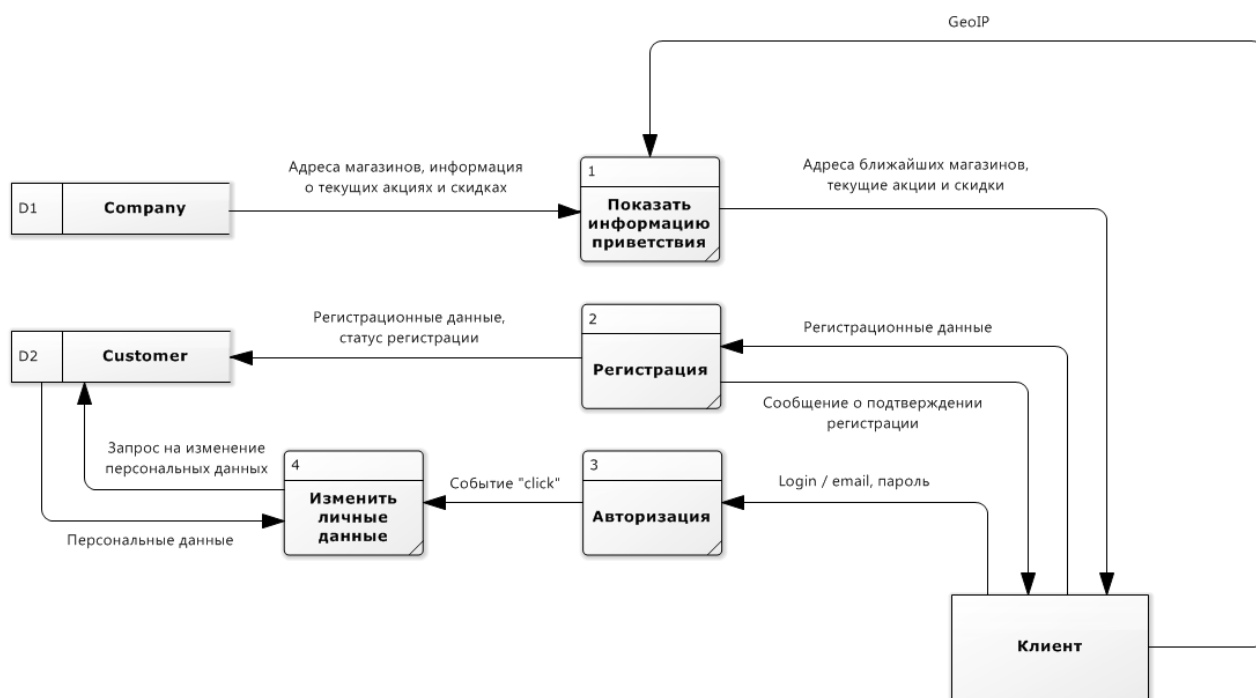


Рисунок 1.6.4 — Диаграмма потоков данных для общих процессов

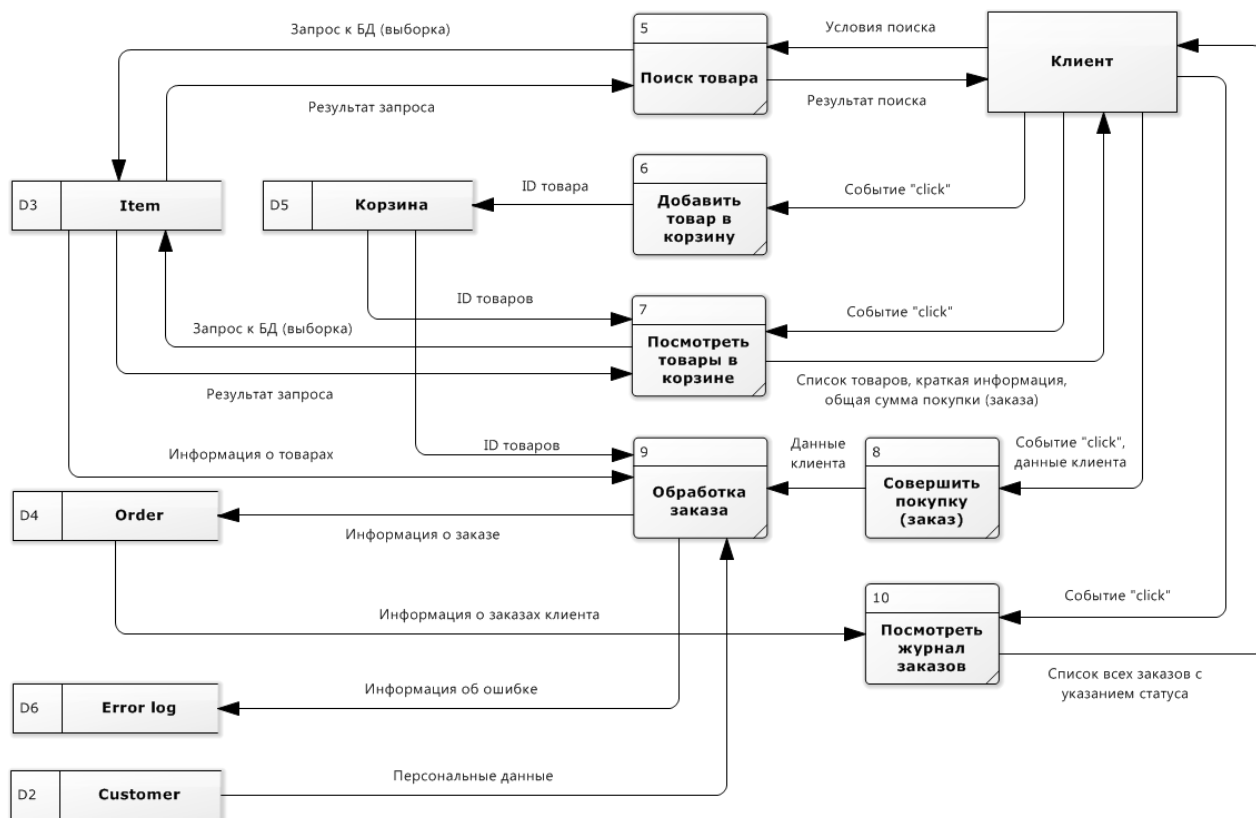


Рисунок 1.6.5 — Диаграмма потоков данных для процессов, имеющих прямое или косвенное отношение к покупке товара

Диаграмма потоков данных 2-го уровня

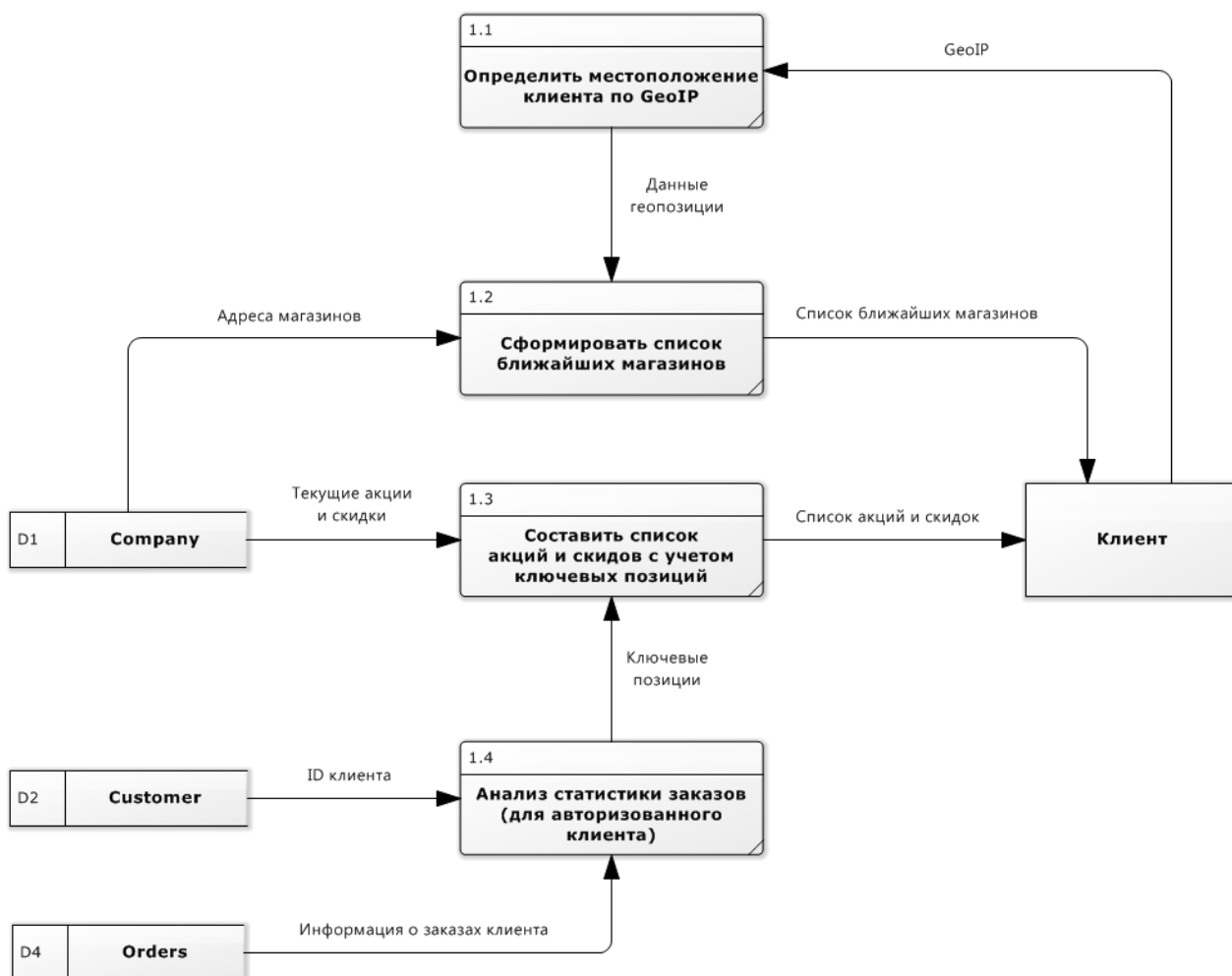


Рисунок 1.6.6 — Декомпозиция процесса №1 «Показать информацию приветствия»



Рисунок 1.6.7 — Декомпозиция процесса №8 «Совершить покупку (заказ)»

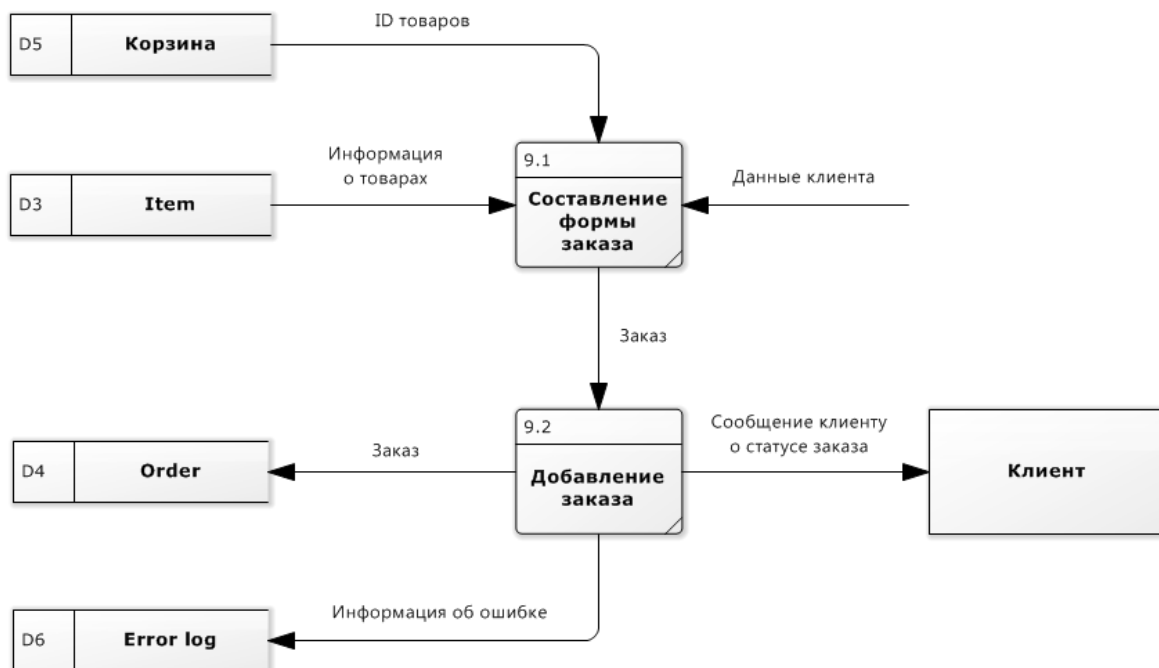


Рисунок 1.6.8 — Декомпозиция процесса №9 «Обработка заказа»

На диаграмме потоков данных 2-го уровня должны быть отражены сущности и накопители данных, с которыми осуществляется взаимодействие исходного процесса на 1-ом уровне иерархии.

Обратите внимание, что некоторые потоки данных не отражены на диаграмме 1-го уровня, например поток «Персональные данные», связывающий накопитель D2 и подпроцесс 8.2 (рис. 1.6.7) или «Сообщение клиенту о статусе заказа» (рис. 1.6.8). Диаграмма потоков данных не ставит перед собой задачу, обозначить совершенно все потоки данных, которые присутствуют в анализируемой или проектируемой системе, кроме того, это ухудшит ее читаемость, поэтому указывают только те потоки, которые существенны на данном уровне иерархии диаграммы и стадии проектирования системы.

При декомпозиции также могут появляться новые хранилища данных (рис. 1.6.8), накопитель D6 на диаграмме 1-го уровня не отражен, так как запись в журнал регистрации ошибок происходит только при возникновении ошибки во время добавления заказа и данный накопитель не является принципиальным для понимания функциональных требований к проектируемой системе.

Спецификация процессов для структурной модели системы «Интернет-магазин»

Таблица 1.6 — Способы представления спецификации:

Структурированный естественный язык	Псевдокод
Процесс 1 Процесс 5 Процесс 6 Процесс 7 Процесс 10	Процесс 2 Процесс 3 Процесс 4 Процесс 8 Процесс 9

Спецификация процессов на структурированном естественном языке:

Процесс 1 «Показать информацию приветствия»

Подпроцесс 1.1 «Определить местоположение клиента по IP»

Подпроцесс 1.2 «Сформировать список ближайших магазинов»

Подпроцесс 1.3 «Составить список акций и скидок с учетом ключевых позиций»

Подпроцесс 1.4 «Анализ статистики заказов (для авторизованного клиента)»

ВХОДНЫЕ ДАННЫЕ: IP клиента; статистические данные

ВЫХОДНЫЕ ДАННЫЕ: список ближайших магазинов, список акций и скидок

ВЫПОЛНИТЬ «определить местоположение клиента по GeoIP»,

«сообщить клиенту его предполагаемое местоположение»,

«попросить подтвердить (или уточнить) местоположение»

ЕСЛИ «клиент подтвердил местоположение» ТО

ВЫПОЛНИТЬ «вывести список ближайших магазинов»

ИНАЧЕ ВЫПОЛНИТЬ «вывести список ближайших магазинов для предполагаемого местоположения через N секунд»

КОНЕЦ ЕСЛИ

ВЫПОЛНИТЬ «получить данные о текущих акциях и скидках»

ЕСЛИ «клиент авторизован» ТО

ВЫПОЛНИТЬ «анализ статистики заказов клиента за последние N (дней)»,

ЕСЛИ «найжены пересечения статистики заказов клиента и текущих акций» ТО

«вывести соответствующие акции и скидки первыми в списке»,

«вывести прочие текущие акции и скидки по дате»

ИНАЧЕ ВЫПОЛНИТЬ «вывести текущие акции и скидки по дате»

КОНЕЦ ЕСЛИ

ИНАЧЕ ВЫПОЛНИТЬ «вывести текущие акции и скидки по дате»

КОНЕЦ ЕСЛИ

Процесс 5 «Поиск товара»

ВХОДНЫЕ ДАННЫЕ: наименование товара; условия фильтра

ВЫХОДНЫЕ ДАННЫЕ: код товара, наименование, описание, характеристики, цена

ЕСЛИ «заполнено поле наименования товара / заданы условия фильтра» ТО

ВЫПОЛНИТЬ «запрос к БД на поиск соответствующей позиции»,

«вывести результат поиска»

ИНАЧЕ «предложить пользователю заполнить формы поиска»

КОНЕЦ ЕСЛИ

Процесс 6 «Добавить товар в корзину»

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID товара

ВЫХОДНЫЕ ДАННЫЕ: список или (обновленные) пиктограммы товаров в корзине

ЕСЛИ «нажата кнопка «добавить в корзину» ТО

 ВЫПОЛНИТЬ «записать код (ID) товара в cookie»,

 «обновить содержание виртуальной корзины с учетом новых покупок»

КОНЕЦ ЕСЛИ

Процесс 7 «Посмотреть товары в корзине»

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID товаров в корзине

ВЫХОДНЫЕ ДАННЫЕ: наименование товаров, описание, цена; общая сумма покупки

ЕСЛИ «нажата кнопка виртуальной корзины» ТО

 ВЫПОЛНИТЬ «запрос к БД по коду (ID) товара из cookie»

 «вывести описание каждой позиции с указанием общей суммы заказа»

КОНЕЦ ЕСЛИ

Процесс 10 «Посмотреть журнал заказов»

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID клиента

ВЫХОДНЫЕ ДАННЫЕ: список заказов, статус заказов

ЕСЛИ «нажата кнопка «журнал заказов» ТО

 ВЫПОЛНИТЬ «запрос к БД по ID клиента»

 «вывести список всех заказов клиента с указанием их статуса»

КОНЕЦ ЕСЛИ

Спецификация процессов на псевдокоде:

Процесс 2 «Регистрация»

ВХОДНЫЕ ДАННЫЕ: login, password, email

ВЫХОДНЫЕ ДАННЫЕ: сообщение о статусе регистрации

```
if(is_set(registration)) {  
    if(filter(entered_login, VALIDATE_LOGIN) AND // проверка корректности логина  
    filter(entered_email, VALIDATE_EMAIL) AND // проверка корректности почты  
    filter(entered_pass, VALIDATE_PASS)) { // проверка корректности пароля  
        // запрос к БД на совпадение с указанными логином и почтой  
        result = mysql_query(SELECT rows FROM users WHERE  
            login = entered_login AND email = entered_email);  
        if(result) { // если найдены совпадения  
            message('Данный логин или почта уже заняты');  
        } else {  
            // запрос на добавление нового пользователя в БД  
            mysql_query(INSERT INTO users (  
                login = entered_login,  
                email = entered_email,  
                pass = entered_pass);  
            send_registration_mail(entered_email); // отправить письмо пользователю  
            message('На указанный вами электронный адрес выслано письмо  
                для активации учетной записи');  
        }  
    } else { message('Проверьте введенные данные'); }  
}  
// если пользователь перешел на страницу активации по уникальной ссылке  
if(activate_page AND user_hash_link) {  
    // запрос к БД на изменение статуса пользователя  
    mysql_query(UPDATE users SET status = enable WHERE hash = user_hash_link);  
    message('Ваша учетная запись активирована');  
}
```

Процесс 3 «Авторизация»

ВХОДНЫЕ ДАННЫЕ: login / email, password

ВЫХОДНЫЕ ДАННЫЕ: сообщение о статусе авторизации

```
if(is_set(authentication)) { // если пользователь ввел данные для авторизации  
    // запрос к БД на совпадение с указанными логином и паролем  
    result = mysql_query(SELECT rows FROM users WHERE  
        email = entered_email AND pass = entered_pass);  
    if(result) { // если найдены совпадения  
        authentication = true; // записываем в cookie, что пользователь авторизован  
    } else {  
        message('Совпадений не найдено.');
```

Процесс 4 «Изменить личные данные»

ВХОДНЫЕ ДАННЫЕ: событие «click»; имя, дата рождения, пол, email, телефон, адрес, аватар, дополнительная информация
ВЫХОДНЫЕ ДАННЫЕ: новые данные в профиле

```
if(is_set(entered_data)) { // если пользователь ввел новые данные
    if(filter(entered_data, VALIDATE_DATA)) { // и они корректны
        // запрос к БД на изменение персональных данных
        mysql_query(UPDATE users SET
            name      = entered_data[name],
            birth_day  = entered_data[birth_day],
            sex        = entered_data[sex],
            email      = entered_data[email],
            phone      = entered_data[phone],
            address     = entered_data[address],
            avatar      = file_name WHERE id = user_id),
            add_info    = entered_data[add_info];
        } else { message('Проверьте корректность введенных данных'); }
    }
```

Процесс 8 «Совершить покупку (заказ)»

Подпроцесс 8.1 «Быстра покупка»

Подпроцесс 8.2 «Покупка через личный кабинет»

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID товаров

ВЫХОДНЫЕ ДАННЫЕ: данные клиента

```
if(authentication) { // если пользователь авторизован
    // обработать заказ с использованием персональных данных пользователя
    order_processing(order = mysql_query(SELECT data FROM customer AND item));
} else {
    // если пользователь не авторизован,
    // но он ввел персональные данные для быстрой покупки
    if(is_set(entered_data)) {
        if(filter(entered_data, VALIDATE_DATA)) { // и данные корректны
            item_name = mysql_query(SELECT item_name FROM items WHERE
            item_id = shopping_cart(item_id));
            create_array order(
                customer,
                address,
                phone,
                email,
                comment,
                item_id,
                item_name);
            order_processing(order); // обработка заказа
        } else { message('Проверьте корректность введенных данных'); }
    }
}
```

Процесс 9 «Обработка заказа»

Подпроцесс 9.1 «Составление формы заказа»

Подпроцесс 9.2 «Добавление заказа»

ВХОДНЫЕ ДАННЫЕ: ID товаров, наименование, цена; данные клиента
ВЫХОДНЫЕ ДАННЫЕ: сообщение о статусе заказа; запись в журнал ошибки (if exception)

```
order_processing(order) {  
    // запрос к БД на добавление нового заказа  
    mysql_query(INSERT INTO orders (  
        date           = get_date(),  
        customer       = order[customer],  
        address        = order[address],  
        phone          = order[phone],  
        email          = order[email],  
        comment        = order[comment],  
        item_id        = order[item_id],  
        item_name      = order[item_name],  
        status         = "pending");  
    unset(shopping_cart); // удалить данные из корзины  
    redirect(home_page); // перенаправить пользователя на домашнюю страницу  
    message('Ваш заказ принят, ожидайте звонка');  
}  
if(exception) { // если произошло исключение (ошибка)  
    exception_handling; // обработать исключение (ошибку)  
    if(error_code) { // если есть код ошибки  
        file_recording(error_log, error_code, add_info); // записать в журнал ошибок  
    }  
}
```

Выводы по работе

В результате выполнения данной лабораторной работы была изучена методология и один из основных инструментов графического структурного анализа — диаграммы потоков данных. Также рассмотрены способы уточнения структурной модели системы (в данном случае диаграммы потоков данных) и получены навыки составления спецификации процессов.

Разработана структурная модель системы «Интернет-магазин» на основе диаграмм потоков данных. Данная модель не описывает весь функционал анализируемой системы, в модели рассматривается только клиентская часть и не затрагивается административная (панель администратора). Также помимо «менеджера (администратора сайта)» за пределы модели вынесены такие сущности, как «склад товаров» и «рекламный отдел», соответственно все процессы, связанные с данными сущностями, в модели не представлены. Для полной декомпозиции контекстной диаграммы потребуется 4 отдельные, связанные между собой диаграммы потоков данных 1-го уровня, что выходит за рамки задания.

Спецификация процессов выполнена двумя способами: на структурированном естественном языке и псевдокоде подобном PHP, стиль псевдокода охарактеризован спецификой информационной системы. Обращения к БД показаны в упрощенной (неточной) форме SQL запросов. В комментариях к псевдокоду между пользователем и клиентом не делается различий и они являются взаимозаменяемыми словами синонимами.

1.7 Контрольные вопросы

- 1) Каково назначение диаграммы потоков данных?
- 2) Что представляет собой внешняя сущность?
- 3) Что по отношению к системе определяют сущности?
- 4) Что представляет собой процесс?
- 5) Является ли процесс атомарным (неделимым) объектом?
- 6) Что представляет собой поток данных?
- 7) Является ли поток данных атомарным (неделимым) объектом?
- 8) Что представляет собой накопитель данных?
- 9) Для чего нужна контекстная диаграмма?
- 10) На каком уровне мы можем прекратить декомпозицию модели?
- 11) Чем отличаются нотации Гейна-Сарсона и Йордана?
- 12) Насколько диаграмма потоков данных должна быть детализирована?
- 13) Что представляет собой спецификация процессов?
- 14) Каково назначение спецификации процессов в структурной модели системы?
- 15) Какие способы представления спецификации процессов допустимы?
- 16) Приведите пример спецификации процесса.
- 17) Какой способ представления спецификации предпочтителен?
- 18) Когда (при каких условиях) составляется спецификация процесса?