РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»

(ФГБОУ ВО РГУПС)

М.А. Бутакова, Т.М. Линденбаум

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Учебно-методическое пособие
для курсовой работы

Ростов-на-Дону
2017

УДК 681.3.06(07) + 06

Рецензенты: доктор технических наук, профессор, А.В. Чернов (РГУПС);
кандидат технических наук, доцент В.В. Ильичева (РГУПС)

Бутакова, М.А.

Технологии программного обеспечения: учебно-методическое пособие для курсовой работы / М.А. Бутакова, Т.М. Линденбаум; ФГБОУ ВО РГУПС. – Ростов н/Д, 2017. – 27 с.

Приведены теоретические и методические рекомендации по теме проектирования и разработки объектно-ориентированных программ, основные сведения о проектировании программ с использованием универсального языка моделирования в рамках рационального унифицированного процесса. Описаны основные этапы проектирования и модели, используемые для решения задач каждого из этапов. Подробно рассмотрена оценка рисков, связанных разработкой программ. Описана методология проектирования по этапам, включающая анализ предметной области, разработку концептуальной модели, поведенческой модели и разработки кода программы. Приводится структура курсовой работы и варианты заданий.

Предназначено для магистрантов направления «Информатика и вычислительная техника», а также для студентов, аспирантов и магистрантов всех специальностей, изучающих дисциплину «Технологии программного обеспечения» и смежные дисциплины.

Одобрено к изданию кафедрой «Информатика».

Оглавление

Цель курсового проектирования

Целью курсовой работы является изучение основных этапов и основных принципов проектирования и разработки объектно-ориентированных программ, практическое освоение методов построения моделей и их использование в процессе разработки.

1 Структура курсовой работы

Дальнейшее изложение методики выполнения работы будет опираться на структуру пояснительной записки (ПЗ) Пояснительная записка к курсовой работе должна содержать следующие разделы, оформленные в соответствии с правилами ЕСПД и [3] :

Содержание

Введение

Задание по лабораторной работе (номер варианта и полное описание своего

варианта).

1. Анализ проблемы

1.1. Словесное описание задачи

1.2. Анализ рисков

1.3. Анализ требований на основе модели прецедентов;

2. Проектирование программы

2.1. Построение модели предметной области

2.2. Уточнение модели прецедентов с использованием системных операций

2.3. Реализация системных операций на основе шаблонов проектирования.

3. Разработка программы.

3.1. Выбор и обоснование среды разработки.

3.2. Разработка интерфейса.

3.3. Контрольный пример.

Заключение.

Список использованных источников.

Приложение.

Реализация основных классов программы на языке программирования.

2 Основные разделы пояснительной записки.

2.1 Введение

Введение является первым пунктом пояснительной записки. Во введении приводятся краткие теоретические сведения об объектно-ориентированной разработке программ и запланированные основные этапы выполнения данной курсовой работы (КР).

2.2 Словесное описание задачи.

На этом этапе проводится анализ предметной области, для которой разрабатывается ПО. Цели этапа:

- определение границ, или контура, системы;
- описание объектов автоматизации и/или формализации знаний
- об этих объектах;
- выявление или определение потребностей заказчика ПО.

Анализ предметной области можно проводить, например, основываясь на теории системного анализа и использовать предложенные в ней методы.

Исходными данными для этапа системного анализа являются:

- регламенты работы отделов и должностные инструкции сотрудников
- этих отделов;
- анкеты опроса заинтересованных лиц;
- записи интервью с заинтересованными лицами;
- другие документы, имеющие отношение к исследуемому объекту.

Выходными данными, или результатом, этапа системного анализа являются:

- перечень заинтересованных лиц;
- список потребностей заинтересованных лиц в разрабатываемом ПО;
- описание объектов автоматизации;
- модель объектов автоматизации или предметной области.

Составление списка заинтересованных лиц

Заинтересованные лица – это все те, кто имеет прямое или косвенное отношение к процессу, автоматизация которого производится.

Для выявления заинтересованных лиц необходимо ответить на следующие вопросы:

- кто является пользователем системы?
- кто является заказчиком (покупателем) системы?
- на кого еще окажут влияние результаты работы системы?
- кто будет оценивать и принимать систему, когда она будет представлена и развернута?
- существуют ли другие внутренние или внешние пользователи системы,
- чьи потребности необходимо учесть?
- кто будет заниматься сопровождением новой системы?
- не забыли ли мы кого-нибудь?

Зафиксируйте не менее трех потребностей или проблем с наивысшими приоритетами, выявленных вами для основных заинтересованных лиц.

На основе этих данных нужно сформулировать перечень потребностей заказчиков, описать модель предметной области и описать объект/объекты автоматизации. Все эти результаты в дальнейшем будут использованы при написании технического задания (ТЗ) на разрабатываемую систему.

Далее необходимо провести аналогию между выявленными потребностями и структурой, и требованиями ТЗ в соответствии с ГОСТ 19.106-78. Таким образом, потребности заказчика в ТЗ могут быть описаны в разделе «Назначение и цели создания системы».

2.3 Анализ рисков.

На этом этапе необходимо выполнить оценку предварительной трудоемкости и стоимости разработки программы.

В оценке стоимости ПО используют две единицы размера: строка кода LineofCode (LOC) [1] и функциональная точка FunctionPoint (FP) [2].

LineofCode – это строка исходного кода ПО (исключаются пустые строки, комментарии и специфические операторы). К преимуществам использования LOC, как единицы размера ПО, относят простоту, а недостатками являются

следующие: размер проекта в LOC может быть определён только после его завершения; LOC зависит от языка программирования; LOC не учитывает качество кода. Производительность (S) программиста с использованием LOC подсчитывается по следующей формуле; где n – количество строк кода, написанных программистом (LOC); m – время работы программиста (в человеко-часах). Видно, что чем больше строк кода, тем выше производительность разработчика. Однако, очевидно можно реализовать одну и ту же функцию, написав меньшее количество строк кода. Единица размера LOC не отражает функциональные свойства кода. Поэтому, если разработчик стремится оптимизировать процесс разработки, с целью уменьшения трудозатрат на реализацию проекта, то при использовании LOC как основной единицы размера проекта под уменьшением трудозатрат подразумевается уменьшение количества строк кода в программе, при этом не оценивается его функциональность.

Рекомендуется в курсовой работе использовать оба подхода: метод функциональных точек FunctionPoint (FP) для получения начальной оценки размера кода на основе оценки функциональности программы и модель раннего этапа разработки EarlyDesignModel (EDM) из семейства моделей COCOMO II для получения окончательных результатов по этапу.

При расчете количества функциональных точек (FP) в соответствии с принятым стандартом используются пять классов компонентов, на которых основывается анализ:

- внутренний логический файл InternalLogicalFile (ILF) – группа логически связанных данных, находящихся внутри границ приложения и поддерживаемых вводом извне;

- внешний интерфейсный файл ExternalInterfaceFile (EIF) – группа логически связанных данных, находящихся вне границ приложения и являющихся внутренним логическим файлом для другого приложения;

- внешний ввод ExternalInput (EI) – транзакция, при выполнении которой данные пересекают границу приложения извне. Это могут быть как данные, получаемые от другого приложения, так и данные, вводимые в программу пользователем. Получаемые данные могут быть командами управления или статическими данными. В последнем случае может возникнуть необходимость обновить внутренний логический файл;

- внешний вывод ExternalOutput (EO) – транзакция, при выполнении которой данные пересекают границу приложения изнутри. Из ILF и EIF создаются файлы вывода или сообщения и отправляются другому приложению. Вывод также содержит производные данные, получаемые из ILF;

- внешний запрос ExternalInquiry (EQ) – транзакция, при выполнении которой происходит одновременный ввод и вывод. В результате информация возвращается из одного или более ILF и EIF. Вывод не содержит производных данных, а ILF не обновляются.

Классы компонентов оцениваются по сложности и относятся к категории высокого, среднего или низкого уровней сложности. Для транзакций (EI, EO, EQ) уровень определяется по количеству файлов, на которые ссылается транзакция FileTypesReferenced (FTR) и количеству типов элементов данных DataElementTypes (DET). Для ILF и EIF имеют значение типы элементов записей RecordElementTypes (RET) и DET. Типы элементов записей — это подгруппа элементов данных в ILF или EIF. Типы элементов данных – это уникальное не рекурсивное поле подмножества ILF или EIF. Уровни сложности и соответствующие им значения FTR и DET описаны в FPCPM [3].

Например, для EI с количеством FTR от 3 и более и DET от 5 до 15 уровень сложности определяется как высокий. Далее компоненты распределяются по «весовым категориям» в зависимости от уровня их сложности. Например, ILF средней сложности имеет значение 10, а EQ высокой сложности значение 6. После этого, производится подсчёт нескорректированных функциональных точек UnadjustedFunctionPoint (UFP) по формуле [1]:

$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} N_{ij} W_{ij}$ где Nij и Wij соответственно количество экземпляров класса i сложности j и его весовое значение. Результат расчёта может быть скорректирован с помощью фактора регулирования стоимости ValueAdjustmentFactor (VAF)[4]. При расчёте VAF учитываются четырнадцать общих характеристиках системы GeneralSystemCharacteristic (GSC), которые оценивают общую функциональность разрабатываемого приложения. Рассчитанное значение количества FP умножается на ориентировочное значение строк кода, которое потребно для реализации одной FP на выбранном языке программирования и в результате получаем значение начальной оценки длины кода программыLOC.

EDM – это высокоуровневая модель, которой требуется сравнительно небольшое количество исходных параметров. Она предназначена для оценки целесообразности использования тех или иных аппаратных и программных средств в процессе разработки проекта. Уравнение модели раннего этапа разработки имеет вид $E = a \cdot LOC^{\wedge}$, где a – константа 2.45. EAF EAF – фактор уточнения затрат (effortadjustmentfactor). Параметры для EDM получаются комбинированием параметров для пост-архитектурной модели. E – искомая трудоемкость разработки программы в человеко-месяцах, которая может быть пересчитана в стоимость.

2.3.1 Анализ требований на основе модели прецедентов.

Термин "прецедент" (usecase) появился в Швеции и в переводе означал "случай использования" (usagecase). Идея описания функциональных требований в виде прецедентов была сформулирована в 1986 году Айваром Якобсоном (IvarJacobson) - главным разработчиком языка UML и UP. Эта идея была признана конструктивной и одобрена широкой общественностью.

Прецеденты — это механизм упрощения этапа формулировки требований для всех заинтересованных лиц. По существу, это рассказы об использовании системы в процессе решения поставленных задач. Основная идея состоит в исследовании и формулировке функциональных требований путем написания историй "из жизни системы". Эти истории помогают сформулировать различные задачи и представляют собой сценарии использования системы.

Исполнителем (actor) будем называть внешнюю сущность, обладающую поведением, например, человека (идентифицируемого по роли), компьютерную систему или организацию, например кассира.

Сценарий (scenario) — это специальная последовательность действий или взаимодействий между исполнителями и системой. Его иногда также называют экземпляром прецедента (usecaseinstance). Это один конкретный сценарий использования системы либо один проход прецедента, например, сценарий успешной покупки товаров за наличный расчет, либо сценарий неудачного завершения покупки из-за прерванной транзакции по обработке данных кредитной карточки.

Неформально, прецедент (usecase) — это набор взаимосвязанных успешных и неудачных сценариев, описывающий использование системы исполнителем для решения одной из задач.

Самый типичный и рекомендуемый тип прецедентов — это прецеденты типа "черный ящик" (black-boxusecases). Они не описывают внутреннюю работу системы, ее компоненты или дизайн. Наоборот, системе вменяются некоторые обязанности (responsibilities). Этот метафорический термин широко применяется в объектно-ориентированном проектировании: программные элементы имеют обязанности и взаимодействуют с другими элементами со своими обязанностями.

Определение основных исполнителей, задач и прецедентов.

Прецеденты предназначены для удовлетворения потребностей основных исполнителей. Поэтому для выделения прецедентов используется следующая процедура:

— Определите рамки системы: является ли она программным приложением, аппаратно-программным комплексом, включает ли в себя своих пользователей или всю организацию?

— Идентифицируйте основных исполнителей, потребности (цели) которых удовлетворяются с помощью системы.

— Для каждого исполнителя определите его задачи. Составьте иерархию

в соответствии с рекомендациями по выделению.

— Определите прецеденты, удовлетворяющие потребности каждого исполнителя, и присвойте им имена в соответствии с задачами.

Степень формализации

Прецеденты описываются в различных форматах, в зависимости от потребностей. Помимо типов "черного ящика" и "белого ящика", выделяют несколько степеней формализации описания прецедентов:

— Сжатый — аннотация в виде одного абзаца. Обычно она описывает только главный успешный сценарий.

— Свободный — неформальный стиль описания. Описание прецедента занимает несколько абзацев и охватывает различные сценарии.

— Развернутый — наиболее подробный стиль описания. При таком подходе детально описываются все шаги и варианты развития сценария, а также предусловия и результаты.

Диаграммы прецедентов.

В языке UML существует система обозначений для диаграммы прецедентов, иллюстрирующей имена прецедентов, исполнителей и взаимосвязи между ними.

2.3.2 Построение модели предметной области

Модель предметной области отображает основные классы понятий (концептуальные классы) предметной области. Она является наиболее важным артефактом, создаваемым на этапе объектно-ориентированного анализа.

Модель предметной области представляет классы понятий реального мира, а не программные компоненты. Это не набор диаграмм, описывающих программные классы или программные объекты с их обязанностями. Следовательно, в модели предметной области не используются артефакты программирования наподобие окон или базы данных, если только разрабатываемая система не является моделью программного средства, например, моделью графического интерфейса пользователя.

На языке UML модель предметной области представляется в виде набора диаграмм классов, на которых не определены никакие операции. Модель предметной области может отображать следующее:

— Объекты предметной области или концептуальные классы.

— Ассоциации между концептуальными классами.

— Атрибуты концептуальных классов.

Идентификация концептуальных классов

Известны два способа выявления концептуальных классов:

— С использованием списка категорий концептуальных классов.

— На основе выделения существительных.

В курсовой работе рекомендуется использовать способ на основе выделения существительных. Он состоит в выделении существительных из текстовых описаний предметной области и их выборе в качестве кандидатов в концептуальные классы или атрибуты.

Для реализации подобного подхода удобно использовать развернутые описания прецедентов.

Представление модели предметной области на языке UML.

Язык UML просто описывает типы диаграмм, например, диаграммы классов или последовательностей. Он не привязан ни к какому методу моделирования или ракурсу моделирования. В отличие от этого, в рамках процесса (например, UP) обозначения языка UML применяются в контексте методологически определенной модели. Рассмотрим некоторые термины, связанные с унифицированным процессом разработки и используемые в UML.

– Концептуальный класс — понятие из реального мира. Рассматривается в концептуальном ракурсе. В рамках UP модель предметной области содержит концептуальные классы.

– Программный класс — класс, представляющий спецификацию или реализацию программного компонента, независимо от процесса или метода.

– Класс проектирования — элемент модели проектирования UP. Это синоним программного класса, однако по определенным причинам автору хотелось акцентировать внимание на том, что данный класс имеет отношение к модели проектирования. В контексте UP классы проектирования относят либо к ракурсу спецификации, либо к реализации.

– Класс реализации — класс, реализованный на объектно-ориентированном языке, например, на Java.

– Класс — в языке UML — это общий класс, представляющий либо понятие реального мира (концептуальный класс), либо программную сущность (программный класс).

Список концептуальных классов предметной области можно представить графически в рамках модели предметной области с использованием языка UML.

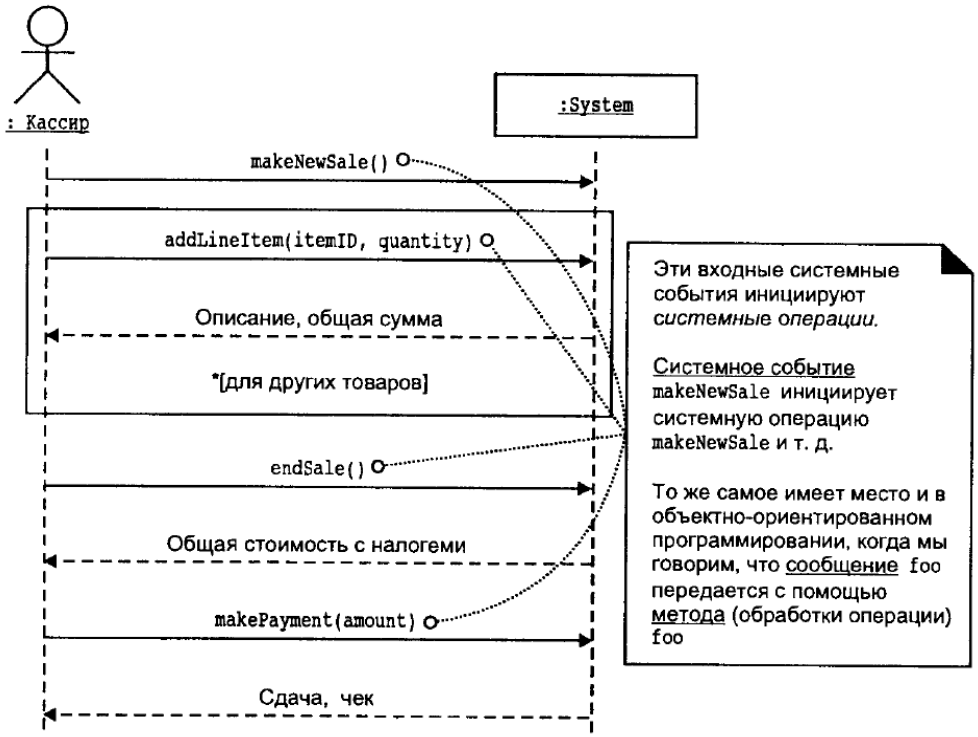
2.3.3 Уточнение модели прецедентов с использованием системных операций

Описания системных операций (systemoperationcontract) описывают детальное поведение системы в терминах изменения состояния объектов модели предметной области после выполнения системных операций.

Системными называются операции, входящие в открытый интерфейс системы для обработки входных системных событий, которые система выполняет как черный ящик. Системные операции можно идентифицировать на основе системных событий.

Весь набор системных операций, выполняемых в процессе всех прецедентов, определяет открытый системный интерфейс, в ракурсе которого систе-

ма рассматривается как единый компонент или класс. В UML систему в целом можно представить в виде одного класса:



Разделы описания системной операции

Операция	Имя операции и ее параметры
Ссылки	(не обязательный) Прецеденты, в рамках которых может выполняться эта операция
Предусловия	Предположения о состоянии системы или объектов модели предметной области до выполнения операции. Выполнение этих условий не проверяется в рамках логики выполнения данной операции, а предполагается, что они истинны. Это нетривиальные условия, на которые читатель должен обратить внимание
Постусловия	Состояние объектов модели предметной области после завершения операции (подробнее обсуждается в следующем разделе)

В разделе "Постусловия" декларируются изменения состояния объектов модели предметной области. К таким изменениям относятся создание экземпляра, формирование или разрыв ассоциации, или изменение атрибута.

Постусловия — это не действия, выполняемые в процессе операции, а лишь декларация об изменении состоянии объектов модели предметной области после выполнения операции.

Существуют следующие категории постусловий описаний.

- Создание и удаление экземпляра.
- Модификация атрибута.
- Формирование и разрыв ассоциаций.

Реже всего встречаются постусловия удаления экземпляра, поскольку

явное разрушение объекта реального мира обычно не имеет значения для разработчиков программной системы. Однако рассмотрим следующий пример. Во многих странах, согласно законодательству, через семь-десять лет после объявления о банкротстве все записи об этом банкротстве должны быть удалены. Заметим, что это относится к концептуальному аспекту, а не к аспекту реализации. Это не означает необходимость очистки памяти компьютера, занимаемой программными объектами.

В процессе составления описаний операций в модель предметной области зачастую приходится вводить новые концептуальные классы, атрибуты или ассоциации. Не привязывайтесь к существующей модели предметной области, усовершенствуйте ее в процессе исследования и описания системных операций.

2.3.4 Реализация системных операций на основе шаблонов проектирования.

Основной задачей этого этапа является создание диаграмм взаимодействия (interactiondiagram), иллюстрирующих взаимодействие объектов в процессе выполнения системных требований. После создания диаграмм взаимодействия (или параллельно с ними) можно построить диаграммы классов, отражающие определения программных классов и интерфейсов, реализованных в программе. Для улучшения качества разрабатываемых диаграмм взаимодействий можно применять систематизированные шаблоны, принципы и идиомы.

Термин "диаграмма взаимодействия" используется в качестве общего названия для двух следующих конкретных типов диаграмм, которые могут использоваться для иллюстрации обмена сообщениями:

- Диаграммы кооперации (collaborationdiagram)
- Диаграммы последовательностей (sequencediagram).

Основные обозначения для диаграмм взаимодействия

Отображение классов и экземпляров объектов.

В языке UML для экземпляра любого элемента языка UML (класса, исполнителя и т.д.) используется то же самое графическое обозначение, что и для типа, однако при этом соответствующая определяющая строка подчеркивается:

Для уникальной идентификации экземпляра класса может использоваться его имя. Если имя экземпляра не указано, на диаграмме кооперации перед именем класса ставится двоеточие (:).

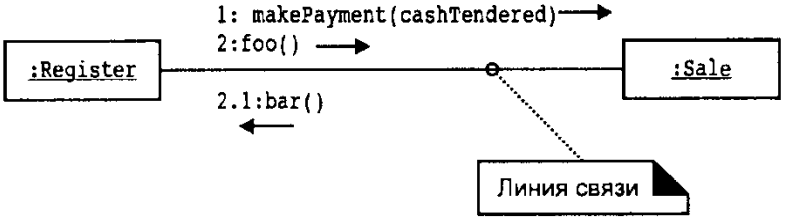
В языке UML существует стандартный синтаксис для обозначения передачи сообщений.

получатель:= сообщение (параметр :типПараметра) : типПолучателя

Информация о типах может исключаться в силу своей очевидности или незначительности. Например:

spec := getProductSpect(id)
spec := getProductSpect(id:ItemID)
spec := getProductSpect(id:ItemlD) : ProductSpecification
Основныеобозначениядиаграммыкооперации
Отображение связей

Связь (link) является соединением между двумя экземплярами классов, определяющим некоторую форму перемещения и видимости между ними. Более строго, можно сказать, что связь является экземпляром ассоциации. Например, имеется связь, или маршрут перемещения, от объекта Register к объекту Sale, в соответствии с которым могут передаваться сообщения, такие как makePayment.



По одной и той же линии связи могут передаваться несколько сообщений в обоих направлениях. Сообщение может передаваться объектом самому себе. Такой случай отображается с помощью связи объекта с самим собой, когда сообщения передаются в направлении этой связи. Для создания экземпляра объекта можно использовать любые сообщения. Однако в языке UML принято использовать для этой цели сообщение create (создать). При использовании другого имени (возможно, менее очевидного) сообщение следует снабдить специальным свойством, получившим название стереотипа UML, в частности <<create>>. Это сообщение передается создаваемому экземпляру объекта.

Представление порядка передачи сообщений

Порядок передачи сообщений иллюстрируется с помощью порядковых номеров (sequencenumber). При этом используется следующая схема нумерации.

1. Первое сообщение не нумеруется.

2. Порядок и вложенность последующих сообщений отображается в соответствии с принятой схемой нумерации. При ее использовании к вложенным сообщениям добавляется номер. Вложенность означает, что к номеру исходящего сообщения добавляется номер входящего сообщения. Условное сообщение изображается с помощью его номера, за которым в квадратных скобках указывается условное выражение, аналогичное условию цикла. Сообщение передается только в том случае, когда оператором возвращается значение true.

Итерационный процесс можно отобразить, указав за порядковым номером сообщения символ *:

2.3.5 Выбор и обоснование среды разработки.

В этом разделе рассматриваются наиболее популярные средства разработки программ и выполняется их сравнительный анализ с точки зрения применения к реализации проектируемой программы.

2.3.6 Разработка интерфейса.

Интерфейс должен быть интуитивно понятным. Таким, чтобы пользователю не требовалось объяснять как им пользоваться.

Для упрощения процесса изучения необходима справка. Буквально — графическая подсказка, объясняющая значение того или иного ЭИ. Полное руководство должно быть частью интерфейса, доступной в любой момент.

Возвращайте пользователя в то место, где он закончил работу в прошлый раз. Зачем нажимать все заново?

Чаще всего, пользователи в интерфейсе сначала ищут сущность(существительное), а затем действие(глагол) к ней. Следуйте правилу «существительное -> глагол». Например, шрифт -> изменить.

Чем быстрее человек увидит результат — тем лучше. Пример — «живой» поиск, когда варианты, в процессе набора поискового запроса. Основной принцип: программа должна взаимодействовать с пользователем на основе наименьшей значимой единицы ввода.

Используйте квазирежимы. Например, ввод заглавных букв с зажатой клавишей shift — это квазирежим. С включенным capslock — режим. Основное отличие в том, что человек может забыть в каком режиме он находится, а в квазирежиме(с зажатой доп. клавишей) это сделать гораздо сложнее.

Простое должно оставаться простым. Не усложняйте интерфейсы. Постоянно думайте о том, как сделать интерфейс проще и понятнее.

Пользователи не задумываются над тем, как устроена программа. Все, что они видят — это интерфейс. Поэтому, с точки зрения потребителя именно интерфейс является конечным продуктом.

Разработка интерфейса обычно начинается с определения задачи или набора задач, для которых продукт предназначен. Для этогоследует использовать сведения из модели прецедентов и описания системных операций.

Какой должен быть дизайн ЭИ?

Тут нужно учитывать все: начиная от цвета, формы, пропорций, заканчивая когнитивной психологией. Однако, несколько принципов все же стоит отметить:

– Цвет. Цвета делятся на теплые(желтый, оранжевый, красный), холодные(синий, зеленый), нейтральные(серый). Обычно для ЭИ используют теплые цвета. Это как раз связано с психологией восприятия. Стоит отметить, что мнение о цвете — очень субъективно и может меняться даже от настроения пользователя.

— Форма. В большинстве случаев — прямоугольник со скругленными углами. Или круг. Полностью прямоугольные ЭИ, лично мне нравятся меньше. Возможно из-за своей «остроты». Опять же, форма как и цвет достаточно субъективна.

— Основные ЭИ(часто используемые) должны быть выделены. Например, размером или цветом.

— Иконки в программе должны быть очевидными. Если нет — подписывайте. Ведь, по сути дела, вместо того чтобы объяснять, пиктограммы зачастую сами требуют для себя объяснений.

Старайтесь не делать слишком маленькие элементы — по ним очень трудно попасть.

Перед расположением, ЭИ следует упорядочить(сгруппировать) по значимости. Т.е. определить, какие наиболее важны, а какие — менее.

Обычно(но не обязательно), элементы размещаются в следующей градации: слева направо, сверху вниз. Слева в верху самые значимые элементы, справа внизу — менее. Это связано с порядком чтения текста. В случае с сенсорными экранами, самые важные элементы, располагаются в области действия больших пальцев рук.

Необходимо учитывать привычки пользователя. Например, если в Windows кнопка закрыть находится в правом верхнем углу, то программе аналогичную кнопку необходимо расположить там же. Т.е. интерфейс должен иметь как можно больше аналогий, с известными пользователю вещами.

Размещайте ЭИ поближе там, где большую часть времени находится курсор пользователя. Что бы ему не пришлось перемещать курсор, например, от одного конца экрана к другому.

Элемент интерфейса можно считать видимым, если он либо в данный момент доступен для органов восприятия человека, либо он был настолько недавно воспринят, что еще не успел выйти из кратковременной памяти. Для нормальной работы интерфейса, должны быть видимы только необходимые вещи — те, что идентифицируют части работающих систем, и те, что отображают способ, которым пользователь может взаимодействовать с устройством.

Делайте отступы между ЭИ равными или кратными друг-другу.

ЭИ должны отвечать. Если пользователь произвел клик, то ЭИ должен как-то отозваться, чтобы человек понял, что клик произошел.

2.3.7 Контрольный пример.

Контрольный пример выполнения программы должен содержать скриншоты экранов, полученные в процессе реализации развернутого сценария прецедента, приведенного в первой главе.

2.4 Заключение

В заключении перечисляются основные этапы выполненной работы по конкретной теме и констатируется степень выполнения каждого этапа.

2.5 Список использованных источников

Содержит перечень учебной, справочной и периодической литературы, а также информационных ресурсов, которые использовались при подготовке КР и на которые есть ссылки в ПЗ.

В приложении приводится код программы на языке реализации.

3 Варианты индивидуальных заданий для курсовой работы:

1. АСУ пассажирскими ж/д перевозками

Описание программы: Автоматизированная система управления пассажирскими железнодорожными перевозками

АСУ включает в себя АРМ (автоматизированное рабочее место) оператора, АРМ кассира и Справочную.

В рамках АРМ оператора реализованы следующие функции:

- учет вагонов, поездов и составов
- формирование пути следования составов
- составление расписания движения составов
- учет проданных билетов

В рамках АРМ кассира реализована функция продажи билетов.

Справочная обеспечивает поиск рейсов для заданных станций отправления и прибытия и заданного диапазона дат отправления, а также получение информации о наличии билетов с разбивкой по классам для заданного рейса.

Реализован удобный интерфейс с пользователем, поиск и фильтрация необходимых данных, формирование отчетов.

Приложение реализовано для платформы клиент-сервер (сервер MS SQL Server).

В базе данных реализованы такие неотъемлемые элементы удаленной БД как хранимые процедуры.

Также в данной работе реализованы связи типа "один-ко-многим" (Master-Detail, "Главный-подчиненный"), SQL-запросы на выборку, модификацию и добавление записей, фильтрация записей БД

2. АЭИС учета материальных ценностей в строительной организации

ОАО "Стройтехника"

Название работы: АЭИС учета материальных ценностей в строительной организации ОАО "Стройтехника"

Описание программы: АЭИС содержит следующие подсистемы и АРМ:

1. склад - 1 АРМ заведующего складом;
2. бухгалтерия - 1 АРМ бухгалтера материального стола;
3. производственный отдел (цех) - 1 АРМ начальника цеха.

АРМ данных подсистем выполняют следующие функции:

- АРМ заведующего складом предназначено для учета документов по поступлению материальных ценностей на склад, регистрации отпуска материальных ценностей производственному отделу по накладным на внутреннее перемещение, формирования сводных документов по поступлению и отпуску материальных ценностей (лимитно-заборная карта, приходный ордер), формирование отчетных документов по движению материальных ценностей (отчет о движении МЦ за месяц, инвентаризационная ведомость);
- АРМ бухгалтера материального стола предназначено для бухгалтерского учета поступления и передачи материальных ценностей в производство, формирования отчетных документов, содержащих полную и достоверную информацию по поступлению и передачи МЦ в строительной организации, обеспечение необходимой информацией внутренних и внешних пользователей бухгалтерской отчетности;
- АРМ начальника цеха предназначено для учета поступлений материальных ценностей по накладным на внутреннее перемещение, расчет остатков МЦ на конец каждого месяца, а также потребности в материальных ценностях в соответствии с планом работ, формирование отчетных документов по движению материальных ценностей (отчет о движении МЦ за месяц, акт о сдаче готовой продукции, калькуляции, инвентаризационная ведомость).

3. АРМ охранника на КПП - база данных "Пропускной пункт предприятия"

Описание программы: АРМ охранника на КПП

В данной работе была спроектирована и реализована система автоматизированного рабочего места охранника КПП.

Данная система обеспечивает следующие основные функции:

- допуск сотрудников и представителей сторонних организаций посредством системы электронного доступа;
- учет въезда/выезда автотранспорта организации и автотранспорта сторонних организаций;
- учет ввоза/вывоза материальных ценностей.

Реализация системы проводилась с использованием BorlandDelphi 6.0. База данных создана в формате *.mdb (MS Access 2000).

4. АРМ предпринимателя: учет компьютерной техники (учет товара и услуг, начисление зарплаты сотрудникам, первичные документы)

Название работы: АРМ предпринимателя: учет компьютерной техники (учет товара и услуг, начисление зарплаты сотрудникам, первичные документы)

Описание программы: АРМ(автоматизированное рабочее место) предпринимателя

Система предназначена для автоматизации работы организации, занимающейся поставкой, наладкой, настройкой компьютерной техники, оказывающей услуги физическим и юридическим лицам по настройке компьютеров, установки ПО и т.д

Автоматизированное рабочее место предпринимателя обеспечивает:

а) ведение базы данных товаров, услуг, контрагентов (клиентов и поставщиков), заключенных договоров, работников организации;

б) учет прихода и расхода товара, товара на складе в наличии;

в) учет оказанных услуг и выполненных работ;

г) начисление зарплаты работникам организации с повременной и сдельной формой оплаты труда;

д) учет движения денег в любой форме (банк, касса);

е) формирование следующих отчетных и первичных документов:

–прайс-лист;

–товар в наличии на складе;

–книга продаж за заданный интервал дат;

–книга покупок за заданный интервал дат;

–платежное поручение;

–счет;

–счет-фактура;

–накладная;

–приходный кассовый ордер;

–акт выполненных работ;

–товарный чек.

Система имеет удобный интерфейс, предоставляющий наиболее гибкий способ просмотра, добавления, редактирования и удаления данных, а также отбор и поиск необходимых данных.

Все документы генерируются в Excel.

Структура БД обеспечивает ведение учета любого товара (т.е. можно вести учет товара и услуг магазина парфюмерии и бытовой химии, канцтоваров, бытовой техники, компакт-дисков, стройматерилов и т.п.).

5. АРМ работника библиотеки (Interbase)

Название работы: АРМ работника библиотеки (Interbase)

Тематика работы: Базы данных

Описание программы: Автоматизируется работа билиотеки. Ведется учет, книг и читателей. Учет книг ведется по разделам, обеспечивается определение книг на руках в данный момент у конкретного читателя и в целом по библиотеке, поиск по различным критериям для книг и читателей. Допускается формирование отчетов.

Система обеспечивает выполнение следующих функций:

-ввод, редактирование и удаление данных о книгах и читателях;

-выдача книги читателю или возврат книги читателем;

-ведение карточки читателя (определение книг на руках у читателя, определение просроченных книг, список книг, которые читатель брал ранее);

-поиск необходимой книги по ее названию, ФИО автора или коду;

-поиск читателя в картотеке по его номер карточки, ФИО или адресу;

-формирование списка – картотеки читателей;

-формирование ведомости выданных книг;

-формирование списка книг по заданному разделу.

6 . "Агенство недвижимости (учет объектов недвижимости и договоров аренды)"

Описание программы: Учет объектов недвижимости и оплаты по ним

Программа должна обеспечивать:

-Ввод, редактирование, хранение и просмотр данных об организациях-клиентах, объектах недвижимости и данных об организации-арендодателе.

-Заключение нового договора, регистрация его в базе данных и последующее хранение, а также формирование типового печатного договора.

-Учет оплаты по договорам, определение должников.

-Формирование и вывод на печать отчетов по доходности недвижимости, объектах недвижимости в наличии, организациях-арендаторах.

В качестве СУБД выбрана MS Access. Для доступа к данным исплользуется технология ADO. Отчеты генерируются в документах Excel.

7. База данных "Автоматизация кафе" (закупка, приготовление, продажа)

Описание программы: Данная программа реально эксплуатировалась в ночном клубе с кухней и двумя барами.

Система автоматизирует весь товарооборот от закупки до регистрации продаж.

В базе есть рецепты(вернее составляющие его) по каждому блюду.

Программа может регистрировать как штучные товары, как бутылка водки, так и составные, как щи.

Регистрация приготовления блюда уменьшает составные на складе и прибавляет готовые на кухне.

Обеспечивается перевод с любого склада(кухни, бара) в любой другой(или списание).

8. База данных "Автовокзал касса"

Описание программы: Программа автоматизация работы кассы автовокзала.

Основная таблица содержит названия остановок, маршрут, время движения до остановки, количество мест в автобусе, цену билета, и количество рейсов в день.

Меню:

Файл(новый, открыть, сохранить, сохранить как, закрыть, печатать, выход)

Правка(вставка записи, удаление, редактирование)

Работа с базой(сортировка, запрос, поиск)

Одна из возможностей запроса: найти остановку (возможно с пересадками) до определённой цены или за определённое время.

9. Книжный магазин

Описание программы: База данных "Книжный магазин"

В реализации базы данных используются связанные таблицы и SQL-запросы.

Программа предназначена для ввода, редактирования, просмотра и хранения данных о книгах в книжном магазине.

Приложение обеспечивает создание ведомостей-отчетов по данным из этой базы.

10. "Табель рабочего времени"

Описание программы: Разработать рабочее приложение для работы с базой данных "Табель учёта рабочего времени сотрудников", данные должны храниться на SQL-сервере Interbase.

Архитектура системе клиент-сервер.

Подготовка тебелей рабочего времени на любой день в Word.

11. "Автосервис"

Описание программы: Программа позволяет вводить и изменять информацию о клиентах, автомобилях, описаний поломок, иметь базу данных

со списком запчастей и их цены , базу данных с ценами на нормочас работы на отдельные части автомобиля, возможность посмотреть историю предыдуших поломок автомобиля.

База данных - paradox.

12. "Перевозки"

Описание программы: Задание:

Построить какую нибудь несложную базу данных(автопарк…) в которой 3 или 4 таблицы.

Сделать вычисляемые поля(calculatedfields), которые вычисляли бы: перевезённый товар по каждому виду транспорта отдельно(в штукак, тоннах), количество потраченного топлива на эти перевозки - общую стоимость (1 литр-0.40 у.е) его за квартал, год, запрос, отчёт, диаграмма.

13. Анализ банковских кредитов (минимизация расходов заемщика при кредитовании)

Описание программы: Программа минимизирует расходы заемщика при кредитовании.

Входные данные: доход заемщика, стоимость квартиры, сумма свободных денег заемщика лдля оплаты оформления кредита и первоначального взноса.

Минимизируется формула затрат заемщика (позже могу предоставить), по 2 переменным (срок кредита и процентная ставка).

Также имеется база данных по банкам (по стоимости оформление кредита и процентным ставкам, информация тоже есть).

На основе входных данных программа выдает предполагаемый банка, где расходы будут минимальны, размере ежемесячного платежа, сумму первоначального взноса, сумму процентов по кредиту и расходов по оформлению кредита, срок кредита.

В программе должна быть возможность добавления новых банков.

Отчеты в Word

splash форма при загрузке

14. АРМ менеджера по продажам в корпоративной системе предприятия (учет продаж компьютерной техники)

Описание программы: Система обеспечивает:

- ведение справочника номенклатуры;
- ведение справочников офисов организации, контрагентов и сотрудников организации;
- формирование счета на продажу и гарантийного талона;

- ведение журнала счетов и журнала заказов;
- создание ценников товара;
- формирование статистики по продажам товара за заданный интервал времени, а также по менеджерам;
- авторизация пользователей и разграничение прав доступа к системе.

Система разработана как клиент-серверное приложение, позволяющее разместить файл БД на сервере, а приложение - на клиенте. Также приложение может работать как обычное локальное (БД + программа на одном компьютере).

Система имеет удобный интерфейс, предоставляющий наиболее гибкий способ просмотра, добавления, редактирования и удаления данных, а также отбор и поиск необходимых данных.

15. АРМ менеджера по продажам в корпоративной системе предприятия (учет продаж мобильных телефонов)

Описание программы: Система обеспечивает:
- ведение справочника номенклатуры;
- ведение справочников офисов организации, контрагентов и сотрудников организации;
- формирование прайс-листа товара;
- формирование счета на продажу и гарантийного талона;
- ведение журнала счетов и журнала заказов;
- создание ценников товара;
- формирование статистики по продажам товара за заданный интервал времени, а также по менеджерам;
- авторизация пользователей и разграничение прав доступа к системе.

Система разработана как клиент-серверное приложение, позволяющее разместить файл БД на сервере, а приложение - на клиенте. Также приложение может работать как обычное локальное (БД + программа на одном компьютере).

Система имеет удобный интерфейс, предоставляющий наиболее гибкий способ просмотра, добавления, редактирования и удаления данных, а также отбор и поиск необходимых
данных.

РЕЦЕНЗИРОВАНИЕ

После проверки курсовой работы преподаватель составляет рецензию. Рецензия должна объективно отражать положительные и отрицательные стороны работы, отличаться деловым стилем, содержать конструктивные замечания.

Объем рецензии зависит от качества выполненной курсовой работы, а также от полноты замечаний и исправлений указанный в рецензии.

Рецензия включает в себя:

- полное наименование вуза;
- заголовок («Карточка рецензента»);
- наименование вида работы;
- наименование дисциплины;
- наименование темы;
- сведения о студенте (фамилия и инициалы);
- сведения о специальности и группе студента;
- дату проверки;
- оценку курсовой работы;
- сведения о преподавателе (фамилия, инициалы), его подпись;
- содержание рецензии.

В рецензии следует отметить:

- характеристика проделанной аналитической и опытно-экспериментальной работы по всем разделам курсовой работы;
- полнота освещения проблемы и раскрытия темы;
- уровень владения технологиями и системами программирования;
- степень самостоятельности, активности и творческой инициативы студента его профессиональные качества;
- техническое и изобразительное качество оформления работы.

Если курсовая работа выполнена на оценку «неудовлетворительно», то работа возвращается студенту с подробными замечаниями для доработки. Курсовая работа выполняется студентом заново.

Студент, курсовая работа которого на дату проведения экзамена не аттестована, не допускается к сдаче экзаменов.

4 Защита курсовой работы

Сроки защиты курсовой работы студентами определяются в соответствии с приказом «Об утверждении графика проведения промежуточной аттестации», утвержденным проректором по учебной работе, с которым деканат (учебный отдел) знакомит студентов на доске объявлений.

Студент, не защитивший курсовую работу в срок, считается имеющим академическую задолженность и не допускается к сдаче экзамена.

Защита курсовой работы осуществляется студентами в установленной устной форме.

Защита курсовой работы проводится в учебной аудитории в присутствии преподавателя, читающего курс, и комиссии, утвержденной приказом ректора. В состав комиссии входят ректор института, проректор по учебной работе, проректор по научной работе, кафедры, руководитель работы.

Защита курсовой работы имеет целью выявить глубину и самостоятельность знаний студента по выбранной теме. На защите студент должен хорошо ориентироваться в представленной работе, уметь объяснять источники цифровых данных, отвечать на вопросы как теоретического, так и практического характера, относящиеся к теме работы.

Перед защитой студент готовится по работе в целом и по замечаниям руководителя.

Защита курсовой работы состоит из краткого изложения студентом основных положений работы. Особое внимание должно быть уделено тем разделам, в которых имеются критические замечания по вопросам избранной темы. В конце сообщения студент отвечает на замечание руководителя, сделанные им в отзыве. После чего члены комиссии задают вопросы. При оценке курсовой работы комиссия учитывает качество написания работы, результаты ее защиты.

5 Литература.

5.1 Кулямин, В.В. Технологии программирования. Компонентный подход / В.В. Кулямин. - М.: Интуит, 2014. - 463 с.

5.2 . Иванова Г.С. Технология программирования : Учебник / Г. С. Иванова. - 3-е изд.,перераб.и доп. - М. : Изд-во МГТУ им.Н.Э.Баумана, 2013. - 336 с. : ил. - (Информатика в техническом университете). - Библиогр.:с.331-333. -Прил.:с.327-330.-Предм.указ.:с.334-335. - ISBN 5-7038-2891-0..

5.3 Плаксин М.А. Тестирование и отладка программ для профессионалов будущих и настоящих / М. А. Плаксин. - М. : БИНОМ, 2013. - 167 с. ISBN978-5-9963-0946-7.http://e.lanbook.com/books/element.php?pl1_id=66305

5.4 Крылов Е.В. Техника разработки программ :Учебник:В 2-х кн. Кн.2 :

Технология, надежность и качество программного обеспечения / Е. В. Крылов, В. А. Острейковский, Н. Г. Типикин. - М. :Высш.шк., 2008. - 470 с. : ил. - Библиогр.:с.463-464. - Прил.:с.432-444.-Глоссарий:с.445-462. -ISBN 978-5-06-005525-2(Кн.2); 978-5-06-005523-8.

5.5 Буч Г.UML. руководство пользователя :/ Г. Буч. - М. : ДМК Пресс, 2008. -496 с. http://e.lanbook.com/books/element.php?pll_id=1246

5.6 Орлов С.А. Технологии разработки программного обеспечения. Современный курс по программной инженерии : учебник для студ. вузов / С. А. Орлов, Б. Я. Цилькер. - 4-е изд. - М. ; СПб. ; Н. Новгород : Питер, 2012. - 608 с. - (Учебник для вузов)

5.7 Ларман К. Применение UML 2.0 и шаблонов проектирования, 3-е издание / Пер. с англ. - М.: «И.Д. Вильямс», 2007. - 736с.: ил.

5.8 Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд. / Пер. с англ. - М.: «Издательство Бином», СПб.: «Невский диалект», 2000. - 560 с.: ил.

5.9 Буч Г., Рамбо Дж., Якобсон А. Язык UML. Руководство пользователя / Пер. с англ. - М.: ДМК Пресс, 2000. - 432 с.: ил. (Серия «Для программистов»)

Учебное издание

Бутакова Мария Александровна
Линденбаум Татьяна Михайловна

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Печатается в авторской редакции
Технический редактор Н.С. Федорова

Подписано в печать 05.10.17. Формат 60×84/16.
Бумага газетная. Ризография. Усл. печ. л. 1,63.
Тираж экз. Изд. № 9071. Заказ .

Редакционно-издательский центр ФГБОУ ВО РГУПС.

Адрес университета:344038, Ростов н/Д, пл. Ростовского Стрелкового Полка Народного Ополчения, д. 2.