Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра Информационных систем и программирования

Дизайн пользовательских интерфейсов

Методические указания по выполнению лабораторных работ для студентов всех форм обучения направления 09.03.03 Прикладная информатика.

Краснодар
2019

Составитель ст. преп. А. А. Ковтун

УДК 004.5

Дизайн пользовательских интерфейсов: методические указания по выполнению лабораторных работ для студентов всех форм обучения направления 09.03.03 Прикладная информатика. / Сост.: А. А. Ковтун; Кубан. гос. технол. ун-т. Каф. Информационных систем и программирования, 2019 – 54 с.

Составлены в соответствии с рабочей программой курса «Дизайн пользовательских интерфейсов» для студентов всех форм обучения направления 09.03.03 Прикладная информатика.

Изложены темы и варианты заданий для выполнения лабораторных работ.

Содержание

№1. Введение. Понятие информационного взаимодействия. Элементы эргономики с точки зрения человеко-машинного взаимодействия.

1 Цели работы

Целями лабораторной работы являются:
1.1 Закрепить теоретические знания по разработке пользовательского интерфейса.
1.2 Получить практические навыки по проведению этапов предварительного и высокоуровневого проектирования интерфейса пользователя.

Задание:
1. Описать приложение из своего варианта по информационному типу взаимодействия с пользователем, описать полностью его интерфейс.
2. На каких уровнях происходит данное взаимодействие.
3. От чего зависит успешное восприятие данного приложения.
4. Составить отчет со скриншотами, пояснениями и выводом.

Краткие теоретические сведения

Современный этап применения компьютерной техники характеризуется следующими чертами:

Во-первых, с распространением персональных компьютеров (ПК) невероятно возросло число пользователей ЭВМ, в том числе не имеющих даже начальных знаний в области вычислительной техники.

Во-вторых, значительно увеличилось и число программирующих пользователей ЭВМ, не имеющих соответствующей базовой подготовки, в их распоряжении имеются мощные средства разработки, которые позволяют создавать программы с практически неограниченными интерактивными возможностями.

В-третьих, имеющиеся государственные стандарты на разработку программных продуктов во многом отстают от потребностей и их влияние на технологию программирования практически невелико.

ИНФОРМАЦИОННОЕ ВЗАИМОДЕЙСТВИЕ

Термин “информация” происходит от латинского “informatio”, что в переводе на русский язык означает “осведомление”, “разъяснение”, и, по сути, предполагает наличие какой-либо формы диалога между отправителями и получателями информации. Все приведенные выше качественные и количественные определения информации также предполагают наличие отправителей и получателей информации, т. е. речь идет о некотором виде взаимодействия объектов. Взаимодействие объектов, приводящее к изменению знаний хотя бы одного из них, будем называть информационным взаимодействием.

Для того, чтобы процесс передачи знаний от одного объекта к другому был успешным, необходимо соблюдение ряда условий. Рассмотрим процесс информационного

взаимодействия на примере передачи знаний посредством устной речи. Процесс этот многокомпонентный (векторный). Первая компонента – физическая, т. е. необходимо наличие физического источника звука (голосовых связок), физической среды распространения звука (воздуха) и физического приемника (уха). Вторая компонента – сигнальная: амплитудно и частотно модулированные колебания. Третья компонента лингвистическая: необходимо, чтобы оба собеседника знали хотя бы один общий язык. Четвертая компонента – семантическая, т. е. в передаваемом сообщении должно присутствовать содержательное описание объекта или влияния, чтобы при получении сообщения могли измениться знания у принимающего эти сообщения. Наконец, пятая компонента – прагматическая: необходимо наличие желания (мотивации) передавать и принимать сообщение.

Следовательно, информационное взаимодействие можно представить пятикомпонентной (пятимерной векторной) величиной, состоящей из компонент:

1. физической (физическое взаимодействие – нажатие клавиш…);

2. сигнальной (сенсорное восприятие сигнала – зрение, слух…);

3. лингвистической (языковое взаимодействие);

4. семантической (смысловая значимость, взаимодействие, реакция на взаимодействие);

5. прагматической (смысловая значимость, полезность, как используется).

Заметим, что приведенное разбиение информационного взаимодействия на пять компонент носит условный характер и возможно частичное пересечение в этом разбиении. Так, отдельные составляющие передаваемого сообщения можно отнести к физической или сигнальной, сигнальной или лингвистической компонентам.

На сложный, многокомпонентный характер информации указывал еще А. Н. Колмогоров :“Подчеркну и качественно новое и неожиданное, что содержится . . . в теории информации. По первоначальному замыслу “информация” не есть скалярная величина. Различные виды информации могут быть чрезвычайно разнообразны . . . было совершенно неясно, можно ли качественно различные информации . . . считать эквивалентными”.

В качестве примера классификации информационных взаимодействий можно напомнить протокольные уровни в международных стандартах открытых компьютерных сетей типа Интернет. При взаимодействии двух пользователей в телекоммуникационной сети реализуется совокупность протоколов семи уровней:

1. физического;
2. канального;
3. сетевого;
4. транспортного;
5. сеансового;
6. представительского;
7. прикладного.

Первые три протокольных уровня определяют такие особенности работы сети связи при обслуживании пользователей, как стандарт электрических сигналов в сети, обнаружение и исправлениеошибок, маршрутизация в транспортной сети и т. д.

Последующие четыре уровня определяют такие стандарты взаимодействия самих пользователей, как контроль за целостностью сообщения, восстановление без потерь сеанса взаимодействия в случае

прерывания, представление данных на дисплеях и печатающих устройствах и т. д. Спектр информационных взаимодействий необычайно широк. Можно условно разделить изучаемые информационные взаимодействия по объектам на три класса:

– 1-й класс – взаимодействие искусственных (технических) систем;
– 2-й класс – взаимодействие смешанных систем;
– 3-й класс – взаимодействие естественных (живых) систем.

К первому классу относятся информационные взаимодействия в технических системах – от простейших регуляторов до глобальных компьютерных сетей.

Ко второму классу – информационные взаимодействия типа “живой организм – искусственный орган”, “человек – машина”, “живой исследователь – неживой объект исследований” и т. д.

К третьему классу относятся информационные взаимодействия, действующие в пределах от молекулярно-генетического уровня до уровня социальных сообществ.

При таком многообразии взаимодействующих объектов задача описания законов информационно-

го взаимодействия необычайно сложна, поскольку надо описать как обмен однобитовой информацией типа “включено – выключено” в технических системах, так и формирование морали в человеческих сообществах.

При описании каждого из этих уровней приходится опираться на специфическую для соответствующего уровня концепцию преобразователя информации, свои языки описания, закономерности,

разрабатываемые в рамках соответствующих дисциплин (наук), которые, тем самым, изучают информационное взаимодействие на данном уровне.

В таблице представлены примеры, относящиеся к описанию информационного взаимодействия в природе и технике для трех типов объектов и пяти компонент информационного взаимодействия.

Несмотря на явно упрощенный характер описания информационного взаимодействия, она может быть полезна для анализа различных определений информации.

Так, энтропийный подход описывает информацию на сигнальном уровне, алгоритмический и алгебраический – на лингвистическом уровне, а логический – на семантическом уровне.Наибольшие успехи были достигнуты при изучении информационного взаимодействия для относительно простых сигнальной и

лингвистической компонент. Из этих компонент удалось сформулировать простые и достаточно общие законы преобразования информации, подобные законам сохранения энергии. Так, Шеннон вывел зависимость скорости передачи информации по каналу с шумом от полосы пропускания, а Колмогоров доказал сохранение сложности при алгоритмических преобразованиях. К сожалению, многочисленные попытки формализованного описания информационного взаимодействия семантической компоненты не привели еще к открытию простых закономерностей, подобно тому, как обстоит дело для сигнальной компоненты.

Для других компонент информационного взаимодействия можно сформулировать сегодня только некоторые принципы – условия, при выполнении которых информационное взаимодействие будет проходить успешно. Для информационного взаимодействия недостаточно только передать сообщение, нужно, чтобы приёмник (адресат) обладал возможностью его адекватно воспринять. Из этого следует принцип тезауруса: важность наличия априорной информации, достаточной для дешифровки и усвоения полученного сообщения. Это означает, в частности, что участники информационного взаимодействия должны обладать согласованной информацией об используемых кодах, языках и их семантиках. Этот принцип подчеркивает первостепенную важность для информатики лингвистических и семантических исследований в широком смысле этого слова. В первую очередь речь идет о фрагментах языков человеческого общения и их семантиках, ставших сегодня основой для разработки средств человеко-машинного диалога.

КОМПОНЕНТЫ ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ	ТИП ОБЪЕКТА		
	ИСКУССТВЕННЫЙ (ТЕХНИЧЕСКИЙ)	СМЕШАННЫЙ	ЕСТЕСТВЕННЫЙ (ЖИВОЙ)
ФИЗИЧЕСКАЯ	Датчики. Сети и каналы связи. Элементы вычислительной техники. Исполнительные устройства.	Вживляемые датчики. Элементы искусственных органов. Физические ос новы сенсомоторики.	Морфология ифизиология рецепторов, эффекторов, нейронов.
СИГНАЛЬНАЯ	Системы автоматического кодирования и декодирования. Системы первичной обработки сигналов.	Искусственные органы. Интерактивные системы первичной обработки информации.	Функции и архитектура нейронных сетей. Первичная обработка сенсорной информации.
ЛИНГВИСТИЧЕСКАЯ	Автоматические системы перевода, программирования, интерпретации и представления сенсомоторной информации.	Формальные грамматики. Языки программирования. Операционные системы. Интерактивные системы работы с базами данных.	Представление знаний в памяти человека и животных. Словарь синтаксис и механизмы работы естественного языка.
СЕМАНТИЧЕСКАЯ	Автоматические системы решения задач, формирования понятий, распознавания образов, логического вывода.	Базы знаний. Экспертные системы. Автоматизированные системы научных исследований, моделирования и проектирования.	Феноменология и принципы ассоциаций, обобщений и умозаключений.
ПРАГМАТИЧЕСКАЯ	Системы формирования и оценки важности целей. Интеллектуальные роботы. Автоматические системы оптимального управления.	Теория управления. Оценки качества управления. Интерактивные системы управления.	Принципы организации поведения человека и животных. Механизмы мотиваций, эмоций, постановки целей и задач.

Успешное восприятие сообщения зависит не только от способности адресата дешифровать (понять) содержание сообщения. Важную роль играет привлекательность сообщения, наличие у адресата стимула для освоения содержания сообщения. Это обстоятельство позволяет сформулировать принцип фасцинации (привлекательности) сообщения, которая зависит от мотивов и целей адресата, формы сообщения и т. д. Важно отметить, что фасцинация характеризует состояние адресата, форму сообщения и, в меньшей степени, его содержание.

Фасцинация как необходимая спутница информации может играть существенную роль в организации человеко-машинного диалога, в проблеме понимания текстов на естественных языках и в других задачах информационного взаимодействия. Следует заметить, что участники информационного взаимодействия в процессе диалога могут выступать не только как получатели информации, но и как ее создатели. Поэтому полезно рассматривать поступающее сообщение не просто как контейнер с готовой информацией, но и как стимул для порождения адресатом на основе прошлого опыта и модели ситуации

мира новой информации – принцип маевтики, или родовспоможения, восходящий к Сократу. В результате такого информационного взаимодействия адресат может “получить”больше информации, чем содержалось в сообщении. Не исключено, что на этом пути будут созданы принципиально новые способы организации человеко-машинного информационного взаимодействия и, возможно, новые структуры систем искусственного интеллекта.

Кроме вышеперечисленных законов и принципов следует упомянуть о разработанных к настоящему времени методах, моделях и алгоритмах информационного взаимодействия, имеющих огромное значение для развития научных исследований. Если говорить об информационном взаимодействии “исследователь – исследуемый объект”, то среди них – методы планирования эксперимента, методы математического и компьютерного моделирования, служащие методологической основой экспериментальных и теоретических исследований.

Популярны у исследователей также методы распознавания образов, идентификации, адаптации, теории массового обслуживания. Этот перечень можно было бы продолжать, однако перечисленное позволяет заключить, что система представлений об информационном взаимодействии сущностей разной природы уже существует, успешно развивается и заслуживает пристального внимания, дальнейшего осмысления и поддержки как новая наука. Следует отметить, что развитие фундаментальных исследований в этой области возможно только при совместной работе специалистов по информатике с математиками, физиками, химиками, биологами, социологами, лингвистами, психологами, т. е. с представителями различных естественных и гуманитарных наук. В то же время, модели и методы информационного взаимодействия будут служить катализаторами развития функциональных исследований природных и социокультурных объектов – от уровня фундаментальных структур материи до уровня взаимодействия социальных групп.

Варианты заданий

1. Стандартное приложение Windows Paint.
2. Стандартное приложение Windows "Калькулятор".
3. Стандартное приложение Windows "Блокнот".
4. Microsoft Word.
5. Windows Media.
6. Web-браузер (на ваш выбор).
7. Проводник.
8. Средство просмотра изображений Windows.
9. Фотоальбом Windows Live.
10. Киностудия Windows Live.
11. Стандартное служебное приложение "Планировщик заданий".
12. Стандартное приложение Windows WordPad.

№2. Типы и разновидности пользовательского интерфейса.

1 Цели работы
1.1 Закрепить теоретические знания по разработке пользовательского интерфейса.
1.2 Развить навыки создания вариантов прототипов интерфейса пользователя.
1.4 Создание собственного интерфейса используя указания и принципы описанные в работе.
1.4 Закрепить теоретические знания по данной теме.

Задание:
1. К какому типу интерфейса относится ваше приложение из л.р. №1. (раздел 2.6)
2. Что относится к элементам данного интерфейса, определить его тип меню, все элементы интерфейса наглядно отобразить на скриншотах.
3. К какому виду интерфейсов относится данное приложение? (раздел 2)
4. Составить отчет с пояснениями и выводом.

Краткие теоретические сведения

Как известно, процесс проникновения информационных технологий практически во все сферы человеческой деятельности продолжает развиваться и углубляться. Помимо уже привычных и широко распространенных персональных компьютеров, общее число которых достигло многих сотен миллионов, становится все больше и встроенных средств вычислительной техники. Пользователей всей этой разнообразной вычислительной техники становится все больше, причем наблюдается развитие двух вроде бы противоположных тенденций. С одной стороны, информационные технологии все усложняются, и для их применения, и тем более дальнейшего развития, требуется иметь очень глубокие познания. С другой стороны, упрощаются интерфейсы взаимодействия пользователей с компьютерами. Компьютеры и информационные системы становятся все более дружественными и понятными даже для человека, не являющегося специалистом в области информатики и вычислительной техники. Это стало возможным прежде всего потому, что пользователи и их программы взаимодействуют с вычислительной техникой посредством специального (системного) программного обеспечения - через операционную систему. Операционная система предоставляет интерфейсы и для выполняющихся приложений, и для пользователей.

1. ПОНЯТИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Интерфейс - совокупность технических, программных и методических (протоколов, правил, соглашений) средств сопряжения в вычислительной системе пользователей с устройствами и программами, а также устройств с другими устройствами и программами.

Интерфейс - в широком смысле слова, это способ (стандарт) взаимодействия между объектами.
Интерфейс в техническом смысле слова задаёт параметры, процедуры и характеристики взаимодействия объектов. Различают:

Интерфейс пользователя - набор методов взаимодействия компьютерной программы и пользователя этой программы.

Программный интерфейс - набор методов для взаимодействия между программами.

Физический интерфейс - способ взаимодействия физических устройств. Чаще всего речь идёт о компьютерных портах.

Пользовательский интерфейс - это совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода / вывода, которые физически обеспечивают связь пользователя и компьютера. Обмен информацией осуществляется передачей сообщения.

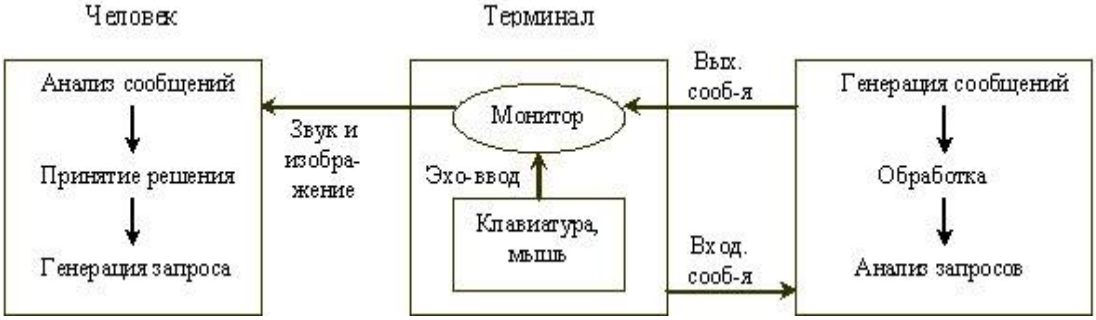


Рисунок 1. Взаимодействие пользователя с компьютером

В основном пользователь генерирует сообщения следующих типов:
запрос информации
запрос помощи
запрос операции или функции
ввод или изменение информации

В ответ пользователь получает подсказки или справки; информационные сообщения, требующие ответа; приказы, требующие действия; сообщения об ошибках и другую информацию.

Интерфейс пользователя компьютерного приложения включает:
- средства отображения информации, отображаемую информацию, форматы и коды;
- командные режимы, язык "пользователь - интерфейс";
- устройства и технологии ввода данных;
- диалоги, взаимодействие и транзакции между пользователем и компьютером, обратную связь с пользователем;
- поддержку принятия решений в конкретной предметной области;
порядок использования программы и документацию на неё.

Пользовательский интерфейс (ПИ) часто понимают только как внешний вид программы. Однако на деле пользователь воспринимает через него всю программу в целом, а значит, такое понимание является слишком узким. В действительности ПИ объединяет в себе все элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением (ПО).

Это не только экран, который видит пользователь. К этим элементам относятся:

- набор задач пользователя, которые он решает при помощи системы;
используемая системой метафора (например, рабочий стол в MS Windows®);

- элементы управления системой;
- навигация между блоками системы;
- визуальный (и не только) дизайн экранов программы;
- средства отображения информации, отображаемая информация и форматы;
- устройства и технологии ввода данных;
- диалоги, взаимодействие и транзакции между пользователем и компьютером;
- обратная связь с пользователем;
- поддержка принятия решений в конкретной предметной области;

- порядок использования программы и документация на нее.

2. ВИДЫ ИНТЕРФЕЙСОВ

Интерфейс - это, прежде всего, набор правил. Как любые правила, их можно обобщить, собрать в "кодекс", сгруппировать по общему признаку. Таким образом, мы пришли к понятию "вид интерфейса" как объединение по схожести способов взаимодействия человека и компьютеров. Вкратце можно предложить следующую схематическую классификацию различных интерфейсов общения человека и компьютера.

Современными видами интерфейсов являются:

1) Командный интерфейс. Командный интерфейс называется так по тому, что в этом виде интерфейса человек подает "команды" компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки. Командная строка — взаимодействие с компьютером осуществляется посредством ввода команд на специальном (машинном) языке в командную строку. Например, в операционных системах ПК.

2) Графический или WIMP - интерфейс (Window - окно, Image - образ, Menu - меню, Pointer - указатель). Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов - меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается "опосредственно", через графические образы. Этот вид интерфейса реализован на двух уровнях технологий: простой графический интерфейс и "чистый" WIMP - интерфейс.

3) SILK - интерфейс (Speech - речь, Image - образ, Language - язык, Knowlege - знание). Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный "разговор" человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера.

4) Жестовый интерфейс — управление с помощью жестов (сенсорный экран, джойстик, руль и т. д.).

5) Нейрокомпьютерный (нейронный) интерфейс — обмен данными между человеческим мозгом и электронным устройством осуществляется с помощью биологической обратной связи и встроенных электронных имплантатов. Например, имитация сетчатки глаза для восстановления зрения.

2.1 Командный интерфейс

Пакетная технология. Исторически этот вид технологии появился первым. Она существовала уже на релейных машинах Зюса и Цюзе (Германия, 1937 год). Идея ее проста: на вход компьютера подается последовательность символов, в которых по определенным правилам указывается последовательность запущенных на выполнение программ. После выполнения очередной программы запускается следующая и т.д. Машина по определенным правилам находит для себя команды и данные. В качестве этой последовательности может выступать, например, перфолента, стопка перфокарт, последовательность нажатия клавиш электрической пишущей машинки (типа CONSUL). Машина также выдает свои сообщения на перфоратор, алфавитно-цифровое печатающее устройство (АЦПУ), ленту пишущей машинки. Такая машина представляет собой "черный ящик" (точнее "белый шкаф"), в который постоянно подается информация и которая также постоянно "информирует" мир о своем состоянии (см. рисунок 1) Человек здесь имеет малое влияние на работу машины - он может лишь приостановить работу машины, сменить программу и вновь запустить ЭВМ. Впоследствии, когда машины стали помощнее и могли обслуживать сразу нескольких пользователей, вечное ожидание пользователей типа: "Я послал данные машине. Жду, что она ответит. И ответит ли вообще? " - стало, мягко говоря, надоедать. К тому же вычислительные центры, вслед за газетами, стали вторым крупным "производителем" макулатуры. Поэтому с появлением алфавитно-цифровых дисплеев началась эра по-настоящему пользовательской технологии - командной строки.

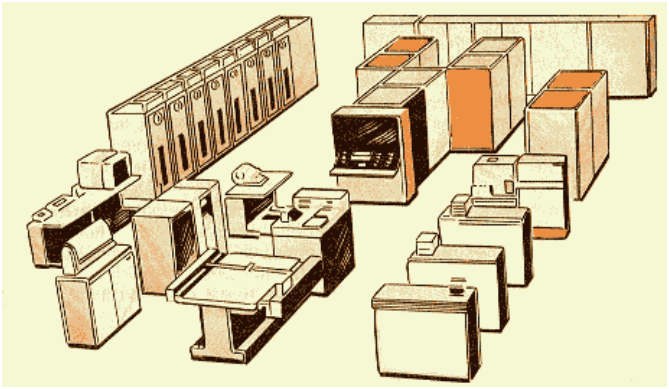


Рис.2. Вид большой ЭВМ серии ЕС ЭВМ

Технология командной строки. При этой технологии в качестве единственного способа ввода информации от человека к компьютеру служит клавиатура, а компьютер выводит информацию человеку с помощью алфавитно-цифрового дисплея (монитора). Эту комбинацию (монитор + клавиатура) стали называть терминалом, или консолью. Команды набираются в командной строке. Командная строка представляет собой символ приглашения и мигающий прямоугольник - курсор. При нажатии клавиши на месте курсора появляются символы, а сам курсор смещается вправо. Это очень похоже на набор команды на пишущей машинке. Однако, в отличие от нее, буквы отображаются на дисплее, а не на бумаге, и неправильно набранный символ можно стереть. Команда заканчивается нажатием клавиши Enter (или Return) После этого осуществляется переход в начало следующей строки. Именно с этой позиции компьютер выдает на монитор результаты своей работы. Затем процесс повторяется. Технология командной строки уже работала на монохромных алфавитно-цифровых дисплеях. Поскольку вводить

позволялось только буквы, цифры и знаки препинания, то технические характеристики дисплея были не существенны. В качестве монитора можно было использовать телевизионный приемник и даже трубку осциллографа.

Обе эти технологии реализуются в виде командного интерфейса - машине подаются на вход команды, а она как бы "отвечает" на них.

Преобладающим видом файлов при работе с командным интерфейсом стали текстовые файлы - их и только их можно было создать при помощи клавиатуры. На время наиболее широкого использования интерфейса командной строки приходится появление операционной системы UNIX и появление первых восьмиразрядных персональных компьютеров с многоплатформенной операционной системой CP / M.

2.2 Графический интерфейс

Как и когда появился графический интерфейс? Его идея зародилась в середине 70-х годов, когда в исследовательском центре Xerox Palo Alto Research Center (PARC) была разработана концепция визуального интерфейса. Предпосылкой графического интерфейса явилось уменьшение времени реакции компьютера на команду, увеличение объема оперативной памяти, а также развитие технической базы компьютеров. Аппаратным основанием концепции, конечно же, явилось появление алфавитно-цифровых дисплеев на компьютерах, причем на этих дисплеях уже имелись такие эффекты, как "мерцание" символов, инверсия цвета (смена начертания белых символов на черном фоне обратным, то есть черных символов на белом фоне), подчеркивание символов. Эти эффекты распространились не на весь экран, а только на один или более символов. Следующим шагом явилось создание цветного дисплея, позволяющего выводить, вместе с этими эффектами, символы в 16 цветах на фоне с палитрой (то есть цветовым набором) из 8 цветов. После появления графических дисплеев, с возможностью вывода любых графических изображений в виде множества точек на экране различного цвета, фантазии в использовании экрана вообще не стало границ! Первая система с графическим интерфейсом 8010 Star Information System группы PARC, таким образом, появилась за четыре месяца до выхода в свет первого компьютера фирмы IBM в 1981 году. Первоначально визуальный интерфейс использовался только в программах. Постепенно он стал переходить и на операционные системы, используемых сначала на компьютерах Atari и Apple Macintosh, а затем и на IBM - совместимых компьютерах.

С более раннего времени, и под влиянием также и этих концепций, проходил процесс по унификации в использовании клавиатуры и мыши прикладными программами. Слияние этих двух тенденций и привело к созданию того пользовательского интерфейса, с помощью которого, при минимальных затратах времени и средств на переучивание персонала, можно работать с любыми программным продуктом. Описание этого интерфейса, общего для всех приложений и операционных систем, и посвящена данная часть.

2.2.1 Простой графический интерфейс

На первом этапе графический интерфейс очень походил на технологию командной строки. Отличия от технологии командной строки заключались в следующим:

1. При отображении символов допускалось выделение части символов цветом, инверсным изображением, подчеркиванием и мерцанием. Благодаря этому повысилась выразительность изображения.

2. В зависимости от конкретной реализации графического интерфейса курсор может

представляться не только мерцающим прямоугольником, но и некоторой областью, охватывающей несколько символов и даже часть экрана. Эта выделенная область отличается от других, невыделенных частей (обычно цветом).

3. Нажатие клавиши Enter не всегда приводит к выполнению команды и переходу к следующей строке. Реакция на нажатие любой клавиши во многом зависит от того, в какой части экрана находился курсор.

4. Кроме клавиши Enter, на клавиатуре все чаще стали использоваться "серые" клавиши управления курсором.

5. Уже в этой редакции графического интерфейса стали использоваться манипуляторы (типа мыши, трекбола и т.п. - см. рис.3) Они позволяли быстро выделять нужную часть экрана и перемещать курсор.

Рис.3. Манипуляторы

Подводя итоги, можно привести следующие отличительные особенности этого интерфейса.

1) Выделение областей экрана.
2) Переопределение клавиш клавиатуры в зависимости от контекста.
3) Использование манипуляторов и серых клавиш клавиатуры для управления курсором.
4) Широкое использование цветных мониторов.

Появление этого типа интерфейса совпадает с широким распространением операционной системы MS-DOS. Именно она внедрила этот интерфейс в массы, благодаря чему 80-е годы прошли под знаком совершенствования этого типа интерфейса, улучшения характеристик отображения символов и других параметров монитора.

Типичным примером использования этого вида интерфейса является файловая оболочка Nortron Commander (о файловых оболочках смотри ниже) и текстовый редактор Multi-Edit. А текстовые редакторы Лексикон, ChiWriter и текстовый процессор Microsoft Word for Dos являются примером, как этот интерфейс превзошел сам себя.

2.2.2 WIMP – интерфейс

Вторым этапом в развитии графического интерфейса стал "чистый" интерфейс WIMP, Этот подвид интерфейса характеризуется следующими особенностями.

1. Вся работа с программами, файлами и документами происходит в окнах - определенных очерченных рамкой частях экрана.

2. Все программы, файлы, документы, устройства и другие объекты представляются в виде значков - иконок. При открытии иконки превращаются в окна.

3. Все действия с объектами осуществляются с помощью меню. Хотя меню появилось на первом этапе становления графического интерфейса, оно не имело в нем главенствующего значения, а служило лишь дополнением к командной строке. В чистом WIMP - интерфейсе меню становится основным элементом управления.

4. Широкое использование манипуляторов для указания на объекты. Манипулятор перестает быть просто игрушкой - дополнением к клавиатуре, а становится основным элементом управления. С помощью манипулятора УКАЗЫВАЮТ на любую область экрана, окна или иконки, ВЫДЕЛЯЮТ ее, а уже потом через меню или с использованием других технологий осуществляют управление ими.

Следует отметить, что WIMP требует для своей реализации цветной растровый дисплей с высоким разрешением и манипулятор. Также программы, ориентированные на этот вид интерфейса, предъявляют повышенные требования к производительности компьютера, объему его памяти, пропускной способности шины и т.п. Однако этот вид интерфейса наиболее прост в усвоении и интуитивно понятен. Поэтому сейчас WIMP - интерфейс стал стандартом де-факто.

Ярким примером программ с графическим интерфейсом является операционная система Microsoft Windows.

2.3 Речевая технология

С середины 90-х годов, после появления недорогих звуковых карт и широкого распространения технологий распознавания речи, появился так называемый "речевая технология" SILK - интерфейса. При этой технологии команды подаются голосом путем произнесения специальных зарезервированных слов - команд. Основными такими командами (по правилам системы "Горыныч") являются:

"Проснись" - включение голосового интерфейса.
"Отдыхай" - выключение речевого интерфейса.
"Открыть" - переход в режим вызова той или иной программы. Имя программы называется в следующем слове.
"Буду диктовать" - переход из режима команд в режим набора текста голосом.
"Режим команд" - возврат в режим подачи команд голосом.

Слова должны выговариваться четко, в одном темпе. Между словами обязательна пауза. Из-за неразвитости алгоритма распознавания речи такие системы требует индивидуальной предварительной настройки на каждого конкретного пользователя.
"Речевая" технология является простейшей реализацией SILK - интерфейса.

2.4 Биометрическая технология
Эта технология возникла в конце 90-х годов XX века и на момент написания книги еще разрабатывается. Для управления компьютером используется выражение лица человека, направление его взгляда, размер зрачка и другие признаки. Для идентификации пользователя используется рисунок радужной оболочки его глаз, отпечатки пальцев и другая уникальная информация. Изображения считываются с цифровой видеокамеры, а затем с помощью специальных программ распознавания образов из этого изображения выделяются команды. Эта технология, по-видимому, займет свое место в программных продуктах и приложениях, где важно точно идентифицировать пользователя компьютера.

2.5 Семантический (общественный) интерфейс

Этот вид интерфейса возник в конце 70-х годов XX века, с развитием искусственного интеллекта. Его трудно назвать самостоятельным видом интерфейса - он включает в себя и интерфейс командной строки, и графический, и речевой, и мимический интерфейс. Основная его отличительная черта - это отсутствие команд при общении с компьютером. Запрос формируется на естественном языке, в виде связанного текста и образов. По своей сути это трудно называть интерфейсом - это уже моделирование "общения" человека с компьютером. С середины 90-х годов XX века публикации, относящихся к семантическому интерфейсу, уже не встречались. Похоже, что в связи с важным военным значением этих разработок (например, для автономного ведения современного боя машинами - роботами, для "семантической" криптографии) эти направления были засекречены. Информация, что эти исследования продолжаются, иногда появляется в периодической печати (обычно в разделах компьютерных новостей).

2.6 Типы интерфейсов

Интерфейсы пользователя бывают двух типов:

1) Процедурно-ориентированные:
Обеспечивают пользователю функции, необходимые для выполнения задач;
Акцент делается на задачи;
Пиктограммы представляют приложения, окна или операции;
Содержание папок и справочников отражается с помощью таблицы-списка.
Включают в себя:

1. Примитивные;
2. Меню;
3. Со свободной навигацией.

2) Объектно-ориентированные:

1. Обеспечивает пользователю возможность взаимодействия с объектами;
2. Акцент делается на входные данные и результаты;
3. Пиктограммы представляют объекты;
4. Папки и справочники являются визуальными контейнерами объектов.

Включают в себя подвид т.н. прямого манипулирования.

Процедурно-ориентированный интерфейс использует традиционную модель взаимодействия с пользователем, основанную на понятиях "процедура" и "операция". В рамках этой модели программное обеспечение предоставляет пользователю возможность выполнения некоторых действий, для которых пользователь определяет соответствие данных и следствием выполнения которых является получение желаемого результата.

Объектно-ориентированные интерфейсы используют модель взаимодействия с пользователем, ориентированную на манипулирование объектами предметной области. В рамках этой модели пользователю предоставляется возможность напрямую взаимодействовать с каждым объектом и инициировать выполнение операций, в процессе которых взаимодействуют несколько объектов. Задача пользователя формулируется как целенаправленное изменение некоторого объекта. Объект понимается в широком смысле слова - модель БД, системы и т.д. Объектно-ориентированный интерфейс предполагает, что взаимодействие с пользователем осуществляется посредством выбора и перемещения пиктограмм соответствующей объектно-ориентированной области. Различают однодокументные (SDI) и многодокументные (MDI) интерфейсы.

Примитивным называется интерфейс, который организует взаимодействие с пользователем и используется в консольном режиме. Единственное отклонение от последовательного процесса, который обеспечивается данными, заключается в организации цикла для обработки нескольких наборов данных.

Интерфейс Меню. В отличие от примитивного интерфейса, позволяет пользователю выбирать операцию из специального списка, выводимого ему программой. Эти интерфейсы предполагают реализацию множества сценариев работы, последовательность действий в которых определяется пользователями. Древовидная организация меню предполагает строго ограниченную реализацию. При этом возможны два варианта организации меню:
каждое окно меню занимает весь экран, на экране одновременно присутствуют несколько разноуровневых меню (Windows).
В условиях ограниченной навигации, независимо от варианта реализации, поиск пункта более чем двух уровневого меню оказывается довольно сложной задачей.

Интерфейс со свободной навигацией (графический интерфейс). Поддерживает концепцию интерактивного взаимодействия с ПО, визуальную обратную связь с пользователем и возможность прямого манипулирования объектом (кнопки, индикаторы, строки состояния). В отличие от интерфейса Меню, интерфейс со свободной навигацией обеспечивает возможность осуществления любых допустимых в конкретном состоянии операций, доступ к которым возможен через различные интерфейсные компоненты ("горячие" клавиши и т.д.). Интерфейс со свободной навигацией реализуется с использованием событийного программирования, что предполагает применение визуальных средств разработки (посредством сообщений).

№3. Метафоры и стили пользовательского ввода: непосредственное манипулирование, выбор из меню, заполнение форм, командный язык, естественный язык.

1 Цели работы

1.1 Закрепить теоретические знания по разработке пользовательского интерфейса.
1.2 Развить навыки создания вариантов прототипов интерфейса пользователя.
1.4 Создание собственного интерфейса используя указания и принципы описанные в работе.
1.4 Закрепить теоретические знания по данной теме.

2 Задание:

2.1 По приложению из 1 лабораторной работы выделить несколько основных способов спецификации интерфейса.
2.2 Какие функции и возможности СУПИ можно определить в данном приложении?
2.3 Перечислите преимущества использования функций СУПИ в вашем приложении.
2.4 Составить отчет со скриншотами, пояснениями и выводом.

МЕТОДЫ И СРЕДСТВА РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Интерфейс имеет важное значение для любой программной системы и является неотъемлемой ее составляющей, ориентированной, прежде всего, на конечного пользователя. Именно через интерфейс пользователь судит о прикладной программе в целом; более того, часто решение об использовании прикладной программы пользователь принимает по тому, насколько ему удобен и понятен пользовательский интерфейс. Вместе с тем, трудоемкость проектирования и разработки интерфейса достаточно велика. По оценкам специалистов в среднем она составляет более половины времени реализации проекта. Актуальным является снижение затрат на разработку и сопровождение программных систем или разработка эффективного программного инструментария.

Одним из путей снижения затрат на разработку и сопровождение программных систем является наличие в инструментарии средств четвертого поколения, позволяющих на высоком уровне описать (специфицировать) создаваемое программное средство и далее по спецификации автоматически сгенерировать исполнимый код.

В литературе не существует единой общепринятой классификации средств для разработки пользовательского интерфейса. Так, программное обеспечение для разработки пользовательского интерфейса можно разделить на две основные группы - инструментарий для разработки пользовательского интерфейса (toolkits) и высокоуровневые средства разработки интерфейса (higher-level development tools). Инструментарий для разработки пользовательского интерфейса, как правило, включает в себя библиотеку примитивов компонентов интерфейса (меню, кнопки, полосы прокрутки и др.) и предназначен для использования программистами. Высокоуровневые средства разработки интерфейса могут быть использованы непрограммистами и снабжены языком, который позволяет специфицировать функции ввода-вывода, а также определять, используя технику непосредственного манипулирования, интерфейсные элементы. К

таким средствам относятся построители диалога (interface builders) и СУПИ - системы управления пользовательским интерфейсом (User Interface Management Systems - UIMS). Помимо СУПИ, некоторые авторы используют такие термины, как User Interface Development Systems (UIDS) - системы разработки пользовательского интерфейса, User Interface Design Environment (UIDE) - среда разработки пользовательского интерфейса и др.

Специализированные средства для разработки интерфейса позволяют упростить разработку пользовательского интерфейса, предлагая разработчику специфицировать компоненты пользовательского интерфейса с использованием языков спецификаций. Можно выделить несколько основных способов спецификации интерфейса:

1. Языковой, когда применяются специальные языки для задания синтаксиса интерфейса (декларативные, объектно-ориентированные, языки событий и др.).

2. Графическая спецификация связана с определением интерфейса, как правило, средствами визуального программирования, программированием демонстраций и по примерам. Подобный способ поддерживает ограниченный класс интерфейсов.

3. Спецификация интерфейса, основанная на объектно-ориентированном подходе, связана с принципом, называемым непосредственное манипулирование. Основное его свойство - взаимодействие пользователя с индивидуальными объектами, а не со всей системой как единым целым. Типичными компонентами, используемыми для манипуляций с объектами и управляющими функциями, являются обработчики, меню, зоны диалога, кнопки различного вида.

4. Спецификация интерфейса по спецификации прикладной задачи. Здесь интерфейс создается автоматически по спецификации семантики прикладной задачи. Однако сложность описания интерфейса затрудняет возможности скорого появления систем, реализующих данный подход.

Основной концепцией СУПИ является отделение разработки пользовательского интерфейса от остального приложения. В настоящее время идея раздельного проектирования интерфейса и приложения либо закреплена в определении СУПИ либо является основным его свойством.

В состав СУПИ определен как набор инструментов этапа разработки и периода исполнения. Инструменты этапа разработки оперируют с моделями интерфейса для построения их проектов. Они могут разделяться на две группы: интерактивные инструменты, например редакторы моделей, и автоматические инструменты, например генератор форм. Инструменты периода исполнения используют модель интерфейса для поддержки деятельности пользователя, например, для сбора и анализа используемых данных.

Функциями СУПИ является содействие и облегчение разработки и сопровождения пользовательского интерфейса, а также управление взаимодействием между пользователем и прикладной программой.

Таким образом, в настоящее время существует большое количество инструментальных средств для разработки интерфейса, поддерживающих различные методы его реализации.

Основное назначение тех средств разработки пользовательских графических интерфейсов,

которые разрабатываются и поставляются отдельно от оконной системы, является облегчение создания нового графического интерфейса за счет использования существующих параметризованных заготовок.
Как видно, в принципе это те же самые идеи, на которых основана объектно-ориентированная библиотека оконной системы X Xt Intrinsics.

И действительно, наиболее распространенный пакет, предназначенный для быстрой и качественной разработки графических пользовательских интерфейсов, Motif, который был спроектирован и разработан в северо-американском консорциуме OSF, в основном является развитием идей Xt Intrinsics. Motif является сугубо коммерческим продуктом. Дело дошло до того, что компания OSF запатентовала внешний интерфейс продуктов, входящих в состав Motif, чтобы не дать кому-нибудь возможность воспроизвести этот интерфейс.

Это привело к настоящему скандалу в сообществе американских программистов, потому что создало опасный прецедент патентования интерфейсов. Если можно закрыть своим авторским правом возможность повторения графического интерфейса, то почему нельзя запатентовать синтаксис языка программирования, интерфейс операционной системы и т.д.? Однако строгость американских законов не оправдывается необязательностью их исполнения, и поэтому недовольные свободолюбивые американские программисты ропчат, но терпят, а тем временем пытаются сагитировать восточно-европейских программистов (не так сильно зависящих от американских законов) на нелегальную свободно доступную реализацию интерфейсов Motif.

С другой стороны, сравнительно недавно (4-5 лет тому назад) в Калифорнийском университете г. Беркли был создан альтернативный механизм под названием Tcl/Tk. Этот механизм основан на наличии специализированного командного языка, предназначенного для описания графических пользовательских интерфейсов, соответствующего интерпретатора и библиотеки ранее разработанных заготовок интерфейсов. Пакет Tcl/Tk распространяется (вместе с полной документацией) свободно, и многие профессиональные программисты находят его более удобным, чем Motif.

№4. Создание прототипа интерфейса и его тестирование

1 Цель лабораторной работы

1.1. Приобретение умений по формированию электронного прототипа – демонстрационного ролика интерфейса.
1.2. Приобретение практических навыков по созданию тестовых заданий.
1.3. Закрепление теоретических знаний и приобретение практических навыков по проведению тестирования интерфейса.

2 Задание на лабораторную работу

2.1. Собрать полную функциональную схему приложения в виде диаграммы последовательности действий.
2.2. Выполнить проверку соответствия структуры полной схемы и последовательностей действий, то есть проверить, нет ли лишних или тупиковых состояний.
2.3. При выявлении несоответствий внести коррективы в содержание экранных форм и/или схему навигации по приложению.
2.4. Повторить предыдущие шаги для откорректированной схемы. Если используется несколько экранных форм, показать их все. Создать наброски в виде диаграмм состояний, в которых могут находиться формы, со стрелками от состояния 1 к состоянию 2, и т.д.
2.5. Сформировать слайды со скриншотами формы или набросками для создания демонстрационного ролика. Каждый слайд соответствует определенному состоянию отдельной экранной формы. (Презентация делается для проектируемого приложения, все состояния форм используются в виде слайдов. На кнопках размещаются элементы без заливки (напр. прямоугольники), с гиперссылками на внутренние слайды. По итогу, получается рабочее приложение, показанное в виде презентации.)
2.6. Согласно полной схеме приложения собрать демонстрационной ролик. Для организации переходов между слайдами использовать гиперссылки (на кнопках смены состояния).
2.7. Для оценки работоспособности системы исходя из пользовательских сценариев подготовить тестовые задания, которые определят, правильно ли работает система. Зафиксировать количественные оценки качества разрабатываемого интерфейса.
2.8. Составить отчет с пояснениями и выводом.

Тестирование интерфейса является исключительно важной задачей при проектировании интерфейса. Начальный этап тестирования связан с разработкой прототипа интерфейса. На этом этапе проектировщик использует имеющиеся результаты проектирования: общую схему приложения, планы отдельных экранных форм, глоссарий. Эти результаты сводятся воедино в общую схему, которую необходимо проверить по сформулированным ранее сценариям. Целью такой проверки является выявление несоответствие последовательности действий, описанной в сценарии, и структуры полной схемы. Обнаруженные несоответствия должны быть устранены за счет модификации экранных форм и/или корректировки общей схемы приложения.
Имея полную схему приложения, приступают к формированию электронного прототипа. Следует отметить, что прототип должен в первую очередь отображать функциональность интерфейса результирующей системы, поэтому его первые версии делают достаточно «примитивными». Последующие версии прототипа могут быть эстетически более совершенными.

Электронный прототип пользовательского интерфейса представляет собой демонстрационный ролик, выполненный в одной из презентационных программ – MS PowerPoint, MS Visio и др. Каждая экранная форма соответствует отдельному слайду, результат нажатия кнопок имитируется переходами между слайдами. Переходы реализуются с помощью организации гиперссылок. Электронная версия прототипа пользовательского интерфейса позволяет тестировать довольно сложные взаимодействия человека с приложением.

Успех тестирования зависит от правильности и корректности постановки задачи тестирования. Тестирование может быть направлено на подтверждение:

производительности действий при использовании продукта. Оценивается по длительности выполнения задач (тестовых заданий) пользователем. Эффективный продукт позволяет увеличить число пользователей, успешно выполняющих задание, в течение ограниченного времени;

полезности продукта. Продукт является полезным, если позволяет снизить количество человеческих ошибок. Полезный продукт позволяет увеличить число пользователей, способных успешно выполнить задание;

простоты обучения. Оценивается по времени тренинга, необходимого для достижения пользователем определенного уровня владения продуктом;

субъективной оценки пользователей. Пользователи оценивают свое отношение к продукту по десятибалльной шкале. Продукт можно считать успешным, если определенная часть пользователей оценила его на 8 и выше баллов.

Тестовые задания, которые в ходе проведения представляют собой задачи для пользователей, формируют исходя из указанных задач тестирования. Основой формулирования тестовых заданий являются пользовательские сценарии.

Тестирование проводится на представителях пользовательской аудитории ранее не знакомых с разрабатываемым продуктом. Уровень опытности тестируемых пользователей должен соответствовать уровню, определенному в профилях конечных пользователей. Считается, что тестирование на одном пользователе позволяет выявить примерно 60% ошибок. Поэтому число тестируемых пользователей, необходимых для проведения одного сеанса зависит от сложности и объема проектируемого продукта. Для «средних» приложений достаточно 4-8 человек. В ходе тестирования:

категорически запрещено прерывать или смущать пользователя;

нельзя внушать тестируемому, что тестируют его;

желательно присутствие разработчиков приложения (программистов), но их роль в тестировании исключительно пассивная.

В качестве методов проведения тестирования могут быть использованы наиболее простые.

1. Наблюдение за пользователем. Пользователю предъявляется тестовое задание, он его выполняет. Действия пользователя фиксируются. Этот метод эффективен при определении неоднозначности элементов интерфейса: любая неоднозначность, как правило, влечет за собой ошибку пользователя. Поскольку действия пользователя фиксируются, обнаружить ошибки при анализе тестов довольно легко.

Кроме того, этот метод подходит для оценки производительности действий пользователя. Для этого необходимо при фиксировании действий замерять время, потребовавшееся пользователю на его выполнение.

2 Комментарии пользователя. Как и при использовании предыдущего метода тестирования, пользователи выполняют тестовые задания. Действия пользователя также фиксируются, кроме того, фиксируются комментарии им своих действий. В дальнейшем комментарии позволяют выявить недостатки реализации конкретных элементов интерфейса - неудачное расположение элементов управления, плохая навигация и т.д. Этот метод можно использовать для оценки полезности продукта, простоты обучения работы с ним, степени субъективного удовлетворения.

Следует отметить, что метод является «нестабильным»: результаты его использования зависят от личных качеств тестируемого пользователя – его разговорчивости, умения последовательно и внятно излагать свои мысли.
3. Качество восприятия. Пользователю предъявляется тестовое задание, через некоторое время после его выполнения, пользователь должен воспроизвести экранные формы (бумажный вариант), с которыми он работал. Результат воспроизведения сравнивают с оригиналом. Идея теста заключается в следующем. Из-за ограничения на объем кратковременной памяти, количество элементов экранных форм, которые запоминает тестируемый, не может быть выше порога запоминания. Пользователь запоминает только то, что считает наиболее актуальным в процессе работы. Следовательно, при повторном выполнении задания пользователю, знающему расположение необходимых для этого элементов интерфейса, будет проще. Таким образом, этот метод позволяет оценить простоту обучения работе с продуктом, а, кроме того, степень субъективной удовлетворенности пользователей.
Следует отметить, что выявление в ходе тестирования различных ошибок и несоответствий неизбежно. Это является одной из причин того, что тестирование нельзя переносить на окончание проекта, когда вносить модификации нет возможности из-за истечения сроков работ.
Разработка пользовательского интерфейса приложения представляется собой итеративный процесс. Каждая итерация связана с отдельным этапом проектирования, созданием прототипа по его результатам, тестированием прототипа и его модификацией. Разработчик должен прилагать особые усилия, чтобы уменьшить число итераций.

Порядок выполнения работы

Вначале выполняют формирование бумажного прототипа интерфейса. Он представлен полной схемой продукта. На этом этапе можно воспользоваться построенным ранее графом состояния главного меню. Каждому состоянию меню соответствует определенная экранная форма приложения. Кроме того, полная схема должна предусматривать отображение навигационной системы продукта в целом, как между экранными формами, так между элементами управления, содержащимися в отдельных формах. Поэтому в полную схему включают изображения форм, соответствующие различным состояниям включенных в них элементов.
Проверка бумажного прототипа на соответствие пользовательским сценариям является своеобразным тестированием. Бумажный прототип имеет значительное преимущество в отношении организации тестирования и модификации прототипа. Модификации могут относиться к проектированию на низком уровне и быть связаны с расположением элементов управления на форме, с изменением последовательности перехода между элементами, с заменой использованных элементов на другие и т.д. Также может оказаться, что выполнить некоторые фрагменты сценариев невозможно, либо затруднительно. Такое обстоятельство потребует возврата к этапу высокоуровневого проектирования. Следует отметить, что внесенные изменения должны быть своевременно отражены в соответствующих темах глоссария.
При переходе к формированию электронного прототипа выделяют различные состояния экранных форм. Состояние формы определяется текущим состоянием содержащихся в ней элементов управления, таких состояний может быть множество. Например, состояния командной кнопки: нейтральное, нажатое, нейтральное с установленным фокусом ввода и т.д.; состояния группы радиокнопок: выбрана первая альтернатива, вторая и др.; состояния списков и других элементов управления. Все допустимые сочетания состояний элементов одной формы представляют собой отдельные состояния этой формы. В презентации – электронном прототипе интерфейса – любому из выделенных состояний экранных форм соответствует отдельных слайд.

Когда совокупность необходимых слайдов сформирована, приступают к сборке презентационного ролика. Ролик не может представлять собой смену слайдов в фиксированной последовательности, потому что при работе с приложением пользователь может задействовать любые доступные на текущий момент элементы интерфейса. В связи с этим переход между слайдами организуют с использованием гиперссылок.
Возможность использования гиперссылок имеется и в презентационной программе Microsoft PowerPoint. Гиперссылка здесь является связью одного слайда с другим слайдом, с произвольным показом, с**одержащим группу слайдов другой презентации, которую планируется показать определенной аудитории**, веб-страницей или файлом. Сама по себе гиперссылка может являться как текстом, так и объектом, таким как рисунок**, который можно разгруппировать на отдельные редактируемые объекты, или файл, являющийся одним объектом (такой как точечный рисунок)**, графика или фигура. При помещении указателя на гиперссылку он отображается в форме руки, показывая, что данный объект можно щелкнуть. Текст, представляющий гиперссылку, подчеркнут и окрашен цветом. Рисунки, фигуры и другие объекты с гиперссылками не имеют дополнительных свойств. Для выделения гиперссылок к объектам можно добавить параметры действий, такие как звук или выделение.
В презентацию можно вставить готовую управляющую кнопку и определить для нее гиперссылки. Управляющие кнопки используются, например, при добавлении кнопок с интуитивными символами для перемещения к следующему, предыдущему, первому или последнему слайду презентации.
При создании гиперссылки на какой-либо объект, отличающийся от слайда, путь к нему задается в виде адреса URL (указывающего протокол (такой как HTTP или FTP) и расположение объекта в Интернете или интрасети, например: http://www.microsoft.com или файл://Имя_компьютера/Общая_папка/ИмяФайла.htm. При создании гиперссылки на страницу или файл, расположенные в локальной файловой системе, гиперссылка отображается как путь к файлу, например C:\Documents and Settings\Имя_Пользователя\Мои документы\файл.xls.
Создаваемые гиперссылки могут содержать абсолютные или относительные связи. В абсолютных ссылках используется точный путь; в случае перемещения файла, содержащего гиперссылку или ее конечный объект, связь между ними разрывается. При использовании гиперссылок с относительными связями можно перемещать файлы, содержащие гиперссылки, и объекты, на которые эти ссылки указывают, без разрыва имеющихся связей. Перемещение файла, содержащего гиперссылку, и конечного объекта этой гиперссылки следует выполнять одновременно.
Для создания гиперссылки на произвольный показ или место в текущей презентации необходимо:
Выбрать текст или объект, который должен представлять гиперссылку.
Нажать кнопку Добавление гиперссылки  .
В области Связь с выбрать значок местом в этом документе.
Если устанавливают связь с другим слайдом в текущей презентации, выбрать из списка слайд, к которому требуется перейти.
Если устанавливают связь с произвольным показом, выбрать из списка произвольный показ, к которому требуется перейти, затем установить флажок показать и вернуться.
Для создания гиперссылки на определенный слайд в другой презентации выполняют следующие действия:
Выбрать текст или объект, который должен служить гиперссылкой.
Нажать кнопку Добавить гиперссылку  .
В области Связать с выбрать пункт имеющимся файлом, веб-страницей.
Найти и выбрать презентацию, содержащую слайд, на который должна указывать гиперссылка.

Нажать кнопку Закладка и выбрать заголовок требуемого слайда.
Создание гиперссылки на файл требует выполнения следующих действий.
Выбрать текст или объект, который должен служить гиперссылкой.
Нажать кнопку Добавить гиперссылку  .
В области Связать с выберите пункт имеющимся файлом.
Перейдите к нужному файлу.
Изменение адреса гиперссылки выполняют следующим образом.
Выбрать гиперссылку.
Нажать кнопку Добавление гиперссылки  .
Выберите нужное назначение.
Для изменения текста гиперссылки сначала указывают необходимый текст гиперссылки, затем в поле вводят новый текст.
При создании презентации со ссылками на ряд файлов рекомендуется поместить эти файлы в общий каталог на сервере и создать базу гиперссылок. Базой гиперссылки является путь, первая часть которого будет общей для файла, содержащего гиперссылку, и для файла, на который она указывает. В случае изменения адреса URL сервера достаточно обновить только базу гиперссылок, а не все пути гиперссылок. Чтобы установить базу гиперссылок необходимо:
Открыть презентацию, для которой следует установить базу гиперссылок.
В меню Файл выбрать пункт Свойства и перейти на вкладку Документ.
В поле База гиперссылок указать путь к файлам для гиперссылок.
Данные для ссылки базы гиперссылок можно переопределить, введя полный адрес гиперссылки в диалоговом окне Вставка гиперссылки.
Удаление гиперссылки без удаления представляющего ее текста или объекта выполняют щелкая правой кнопкой мыши текст или объект, представляющий гиперссылку и выбирая в контекстном меню Удалить гиперссылку. Если требуется удалить и гиперссылку и представляющий ее текст, то сначала объект и текст выделяют, а затем нажимают клавишу DEL.
Важно помнить, что в PowerPoint гиперссылки становятся активными при запуске презентации, а не при ее создании.
Перед представлением презентации следует обязательно проверить ее на наличие гиперссылок с разорванными связями и проверить все гиперссылки на внешние источники. Щелчок гиперссылки с разорванной связью в PowerPoint приводит к сообщению об ошибке. Причиной может являться опечатка в адресе URL или гиперссылка на перемещенный или удаленный объект.
После того как будет сформирован демонстрационный ролик, от проектировщика интерфейса требуется повторно проверить соответствие переходов между слайдами и последовательностей действий, указанных в пользовательских сценариях. Как правило, сначала формируют тестовые задания на основе пользовательских сценариев и необходимую проверку выполняют уже по ним.
Тестовое задание включает последовательность действий записанных в сценарии, но в отличие от него содержит конкретные значения данных, с которыми оперирует пользователь. Рассмотрим пример пользовательского сценария.
Анна Сергеевна общаясь с клиентами по телефону, создает новые заказы. При формировании нового заказа, она выбирает клиента из списка, если его там нет, то вводит клиента в список клиентов. Затем добавляет в заказ необходимые товары, используя сложный поиск. Она распечатывает информацию заказа, после этого она сохраняет ее.
В соответствии со сценарием необходимо имитировать ввод данных клиента при оформлении нового заказа. Имитировать ввод с клавиатуры для прототипа в виде презентации невозможно. Если же данные клиента вводятся из списка, то это достаточно просто. От слайда с формой нового (пустого) заказа организуют переход к слайду со

списком клиентов. Перемещение по списку записей с данными клиентов имитируют последовательным переходом к слайдам с изображением списка со смещенными записями. После указания нужной записи организуют переход к слайду с формой нового заказа с заполненными данными клиента.
Задание атрибутов товара при организации поиска имитируют переходом к слайдам с изображениями списков значений атрибутов. Затем выполняют переход к слайду, отображающему результаты поиска. Если результат одиночный, то от этого слайда организуют возврат к форме с текущим (новым) заказом, если результаты множественные выбор нужной записи имитируют аналогично рассмотренному выше выбору записи из списка.
Для добавления следующего товара используются либо слайды с множественными результатами последнего поиска, либо слайды с изображением атрибутов товара для имитации нового поиска. Когда заданные товары будет добавлены в заказ, используют слайды с изображением выбора команд меню печати и сохранения заказа.
Тестовое задание может быть сформулировано в следующем виде:
Создать новый заказ для клиента ООО Регионторг. Ввести в заказ данные клиента из списка клиентов. Затем организовать поиск требуемого клиенту товара, определяя атрибуты товара: категория – выключатель, цвет – белый, производитель – Польша. Используя результаты поиска, добавить в заказ сначала товар VIKO, затем товар VIKO-2. Организовать новый поиск товара с атрибутами: категория – распределительный щиток, цвет – белый, производитель – Германия. Добавить в заказ товар SIMENS. Распечатать заказ и сохранить его.
Проверка соответствия тестового задания и последовательности перехода между слайдами выполняется на готовом ролике. Выявленные несоответствия могут потребовать изменения навигационной системы приложения.
Когда сформулированы необходимые тестовые задания, выполнены проверки и требуемые коррективы, приступают к тестированию с привлечением пользователей из целевой аудитории.
В начале этого этапа формулируют задачи тестирования. Например, оценить производительность действий при использовании продукта. Тестирование проводится путем наблюдения за пользователем с фиксированием длительности выполнения действий. Критерий оценки можно сформулировать как выполнение контрольного тестового задания в течение 3 минут 75% тестируемых пользователей после тренинга - выполнения пяти различных сценариев. Результаты тестирования анализируются с точки критерия оценки, в качестве которого может выступать, например, выполнение контрольного тестового задания в течение 3 минут 75% тестируемых пользователей после тренинга - выполнения пяти различных сценариев.
Все результаты тестирования обобщаются с тем, чтобы сформулировать рекомендации относительно модификации прототипа интерфейса. Модификации могут быть связаны с изменениями содержимого экранных форм, элементов навигационной системы, терминологии и даже функциональности тех или иных элементов интерфейса.

№5. Разработка пользовательского интерфейса: этапы предварительного и высокоуровневого проектирования

1 Цели работы
Целями лабораторной работы являются:
1.1 Закрепить теоретические знания по разработке пользовательского интерфейса.
1.2 Получить практические навыки по проведению этапов предварительного и высокоуровневого проектирования интерфейса пользователя.

Задание
2.1. Разработать главное меню своего приложения, составить его схему.
2.2. Выполнить этапы предварительного и высокоуровневого проектирования при разработке пользовательского интерфейса приложения для предметной области, соответствующей варианту задания. Создать программу.
2.3. Предусмотреть использование профилей потенциальных пользователей программного обеспечения, т.е. различные состояния программы для различных пользователей.
2.4. Составить отчет с пояснениями и выводом.

2 Краткие теоретические сведения
На практике высокоуровневое проектирование пользовательского интерфейса предваряет первоначальное проектирование, которое позволяет выявить требуемую функциональность создаваемого приложения, а также особенности его потенциальных пользователей. Указанные сведения можно получить, анализируя информацию, поступающую от пользователей. С этой целью производят опрос целевой аудитории и формируют профили пользователей. Профилями называют описания главных категорий пользователей. Одна из таких категорий может быть принята за основной профиль. Следует отметить, что набор характеристик, подробно описывающий пользователя, зависит от предметной области и контекста решаемых им задач. Поэтому работа по определению целей и задач пользователей и работа по формированию их профилей ведется параллельно.
Наиболее общий шаблон профиля содержит в себе следующие разделы:
социальные характеристики;
навыки и умения работы с компьютером;
мотивационно-целевая среда;
рабочая среда;
особенности взаимодействия с компьютером (специфические требования пользователей, необходимые информационные технологии и др.).
Профили пользователей могут по необходимости расширяться за счет добавления других (значимых с точки зрения проектировщика) характеристик пользователей.
После выделения одного или нескольких основных профилей пользователей и после определения целей и задач, стоящих перед ними, переходят к следующему этапу проектирования. Этот этап связан с составлением пользовательских сценариев. Как правило, начинают с персонификации профилей (присваивания каждому профилю условного имени), затем формулируют сценарии. Сценарий - это описание действий, выполняемых пользователем в рамках решения конкретной задачи на пути достижения его цели. Очевидно, что достигнуть некоторой цели можно, решая ряд задач. Каждую их них пользователь может решать несколькими способами, следовательно, должно быть

сформировано несколько сценариев. Чем больше их будет, тем ниже вероятность того, что некоторые ключевые объекты и операции будут упущены.
В то же время, у разработчика имеется информация, необходимая для формализации функциональности приложения. А после формирования сценариев становится известным перечень отдельных функций. В приложении функция представлена функциональным блоком с соответствующей экранной формой (формами). Возможно, что несколько функций объединяются в один функциональный блок. Таким образом, на этом этапе устанавливается необходимое число экранных форм. Важно определить навигационные взаимосвязи функциональных блоков. На практике установлено наиболее подходящим число связей для одного блока равное трем. Иногда, когда последовательность выполнения функций жестко определена, между соответствующими функциональными блоками можно установить процессуальную связь. В этом случае их экранные формы вызываются последовательно одна из другой. Такие случаи имеют место не всегда, поэтому навигационные связи формируются либо исходя из логики обработки данных с которыми работает приложение, либо основываясь на представлениях пользователей (карточная сортировка). Навигационные связи между отдельными функциональными блоками отображаются на схеме навигационной системы. Возможности навигации в приложении передаются через различные навигационные элементы.
Основным навигационным элементом приложения является главное меню. Роль главного меню велика еще и потому, что оно осуществляет диалоговое взаимодействие в системе «пользователь-приложение». Кроме того, меню косвенно выполняет функцию обучения пользователя работе с приложением.
Формирование меню начинается с анализа функций приложения. Для этого в рамках каждой из них выделяют отдельные элементы: операции, выполняемые пользователями, и объекты, над которыми осуществляются эти операции. Следовательно, известно какие функциональные блоки должны позволять пользователю осуществлять какие операции над какими объектами. Выделение операций и объектов удобно проводить на основе пользовательских сценариев и функционала приложения. Выделенные элементы группируются в общие разделы главного меню. Группировка отдельных элементов происходит в соответствии с представлениями об их логической связи. Таким образом, главное меню может иметь каскадные меню, выпадающие при выборе какого либо раздела. Каскадное меню ставит в соответствие первичному разделу список подразделов.
Одним из требований к меню является их стандартизация, целью которой выступает формирование устойчивой пользовательской модели работы с приложением. Существуют требования, выдвигаемые с позиций стандартизации, которые касаются места размещения заголовков разделов, содержания разделов часто используемых в разных приложениях, формы заголовков, организации каскадных меню и др. Наиболее общие рекомендации стандартизации следующие:
группы функционально связанных разделов отделяют разделителями (черта или пустое место);
не используют в названиях разделов фраз (желательно не больше 2 слов);
названия разделов начинают с заглавной буквы;
названия разделов меню, связанных с вызовом диалоговых окон заканчивают многоточием;
названия разделов меню, к которым относятся каскадные меню, заканчивают стрелкой;
используют клавиши быстрого доступа к отдельным разделам меню. Их выделяют подчеркиванием;
допускают использовать «горячие клавиши», соответствующие комбинации клавиш отображают в заголовках разделов меню;
допускают использовать включение в меню пиктограмм;
измененным цветом показывают недоступность некоторых разделов меню в ходе работы с приложением;

допускают делать недоступные разделы невидимыми.
Недоступность некоторых разделов меню обуславливается следующим. Главное меню является статическим и присутствует на экране в течение всего времени работы с приложением. Таким образом, при работе с разными экранными формами (взаимодействии с разными функциональными блоками) не все разделы меню имеют смысл. Такие разделы принято являются недоступными. Поэтому в зависимости от контекста решаемых пользователем задач (иногда от контекста самого пользователя) главное меню приложения выглядит различным образом. О подобных различающихся внешних представлениях меню принято говорить как о различных состояниях меню. В отличие от схемы навигационной системы, составленной ранее и необходимой, в основном, разработчику, с меню пользователь входит в непосредственное взаимодействие. Поэтому следует составить граф состояния меню. Вершинами этого графа являются различные состояния меню (внешние представления одного и того же меню с доступными и недоступными разделами). Каждая вершина имеет пояснения о соответствии данного состояния меню отдельным экранным формам. Дуги графа состояний соответствуют операциям (командам меню), переводящим его из одного состояние в другое.
Подобный граф используют при формировании тестовых заданий на последних стадиях проектирования интерфейса. В связи с этим, важно при его формировании выполнить проверку соответствия пользовательских сценариев возможным переходам по графу.

Порядок выполнения работы

Пусть предметная область представлена информационной системой, отображающую деятельность мелкой фирмы, которая связана с изготовлением и/или поставкой ряда товаров.
Потенциальными пользователями приложения являются, например, менеджеры по направлению товара, торговые представители, представители обслуживающего персонала.Примерные профили некоторых из названных категорий пользователей могут выглядеть следующим образом (таблица 1).
Таблица 1

Пользователи	Менеджер по направлению товара	Представители обслуживающего персонала
Социальные характеристики	Мужчины, женщины Взрослые Русскоязычные Средний уровень владения компьютером	Женщины Взрослые Русскоязычные Низкий уровень владения компьютером
Мотивационно целевая среда	Прямая производственная необходимость, удобство Мотивация к обучению высокая	Производственная необходимость, Престиж Мотивация к обучению низкая
Навыки и умения	Должны иметь значительный тренинг работы с программой	Прошли предварительный тренинг работы с программой
Требования к ПО ИС	Возможность использования ПО ИС в локальной сети Отсутствие жестких ограничений по времени Обеспечение текущей информацией по содержанию заказов	Возможность использования программы одновременно с телефонным общением с клиентом Время реакции ПО ИС, допустимое для ожидания клиента Обеспечение текущей информацией по содержанию заказов

	Обеспечение текущей информацией по товарам Возможность проводить обобщение информации по заказам	Обеспечение текущей информацией по товарам Возможность формирования новых заказов
Задачи пользователя	Просмотр/фильтрация информации по заказам/клиентам/товарам Сортировка информации по заказам/клиентам/товарам Агрегирование информации по заказам/клиентам/товарам	Просмотр данных по товарам Создание/поиск/модификация заказа Сохранение/печать заказа Формирование счета по заказу
Рабочая среда	Стандартизированные ПК, локальная сеть	Стандартизированные ПК, специализированное телефонное обслуживание

Определить функциональность приложения, исходя из целей и задач пользователей.
Рассмотрим определение функциональности на примере одного из профилей: представители обслуживающего персонала. Исходя из задач этой категории пользователей, можно сформировать следующий перечень функций необходимых в приложении:
создать новый заказ (1);
сложный поиск заказа (2);
редактирование заказа (3);
добавление клиента из списка клиентов в заказ (4);
ввод/редактирование клиента в списке клиентов (5);
выбор товара из списка товаров (6);
сложный поиск товаров в списке товаров (7);
просмотр подробных данных о товаре (8);
добавление товара из списка товаров в заказ (9);
сохранение заказа (10);
печать заказа (11);
формирование счета (12).
Сформировать множество пользовательских сценариев для выделенных профилей пользователей.
Примером могут служить приведенные ниже сценарии действий пользователей.
Анна Петровна общается с клиентами по телефону. По просьбе клиента она предварительно просматривает данные о запрошенных им товарах, затем приступает к формированию нового заказа. Она вводит данные клиента, после чего выбирает указанный(ые) товар(ы) из списка и добавляет его (их) в заказ и сохраняет заказ.
Анна Сергеевна общаясь с клиентами по телефону, создает новые заказы. При формировании нового заказа, она выбирает клиента из списка, если его там нет, то вводит клиента в список клиентов. Затем добавляет в заказ необходимые товары, используя сложный поиск. Она распечатывает информацию заказа, после этого она сохраняет ее.
Анна Михайловна выполняет поиск указанного заказа по данным клиента. Она просматривает и при необходимости редактирует данные клиента, добавляет в заказ новые или удаляет из заказа прописанные там товары, при необходимости редактирует в заказе информацию по некоторым товарам, сохраняет информацию и формирует счет заказа.
Анна Николаевна просматривает данные о товаре, выполняет поиск заказа по товару, редактирует в заказе информацию по некоторым товарам, сохраняет информацию и распечатывает ее.

Определить функциональные блоки приложения, составить схему навигационной системы.
Очевидно, что отдельные функциональные блоки соответствуют работе пользователей с информацией:
по заказам (функции 1,2,3,4,9,10,11,12): по общему журналу заказов и по конкретному (текущему) заказу;
по клиентам (функции 4,5): по списку клиентов в целом и по конкретному клиенту;
по товарам (функции 6,7,8,9): по списку товаров и по данному товару подробно.
Таким образом, можно вести речь о наличии в приложении трех функциональных блоков и шести экранных форм:
Журнал заказов;
Текущий заказ;
Список клиентов;
Карта клиента;
Список товаров;
Карта товара.
В этом случае, с учетом пользовательских сценариев схема навигации по формам может выглядеть следующим образом (рисунок 1).

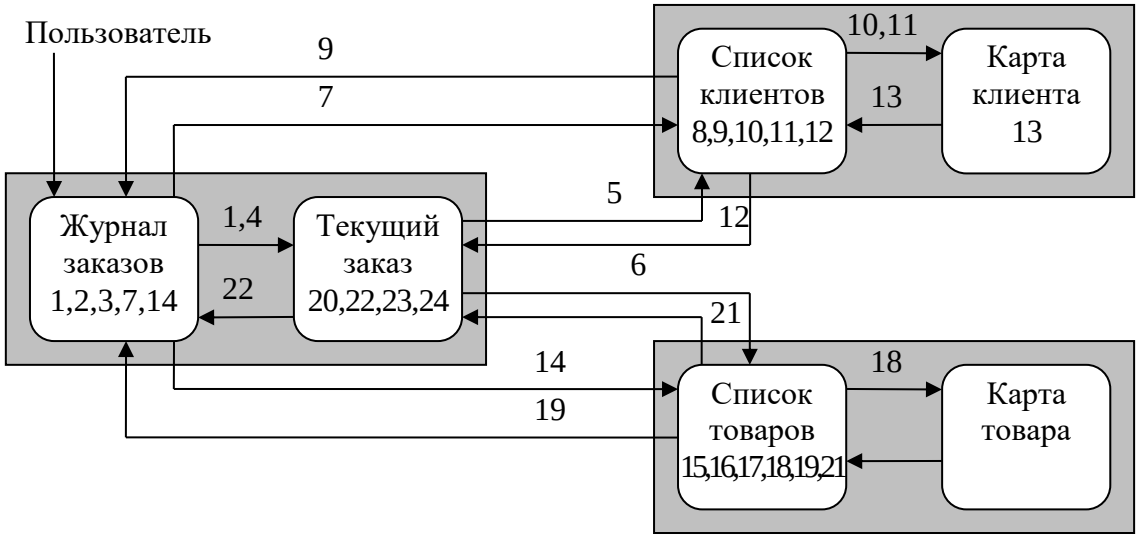


Рисунок 1 – Схема навигации

Цифрами на рисунке обозначены отдельные операции, выполняемые пользователями (п.4.5).
Установить для отдельных функциональных блоков соответствующие им операции и объекты операций. Сгруппировать их в разделы меню. В конкретной среде разработки приложения сформировать меню.
Определим операции, которые должен выполнять пользователь в рамках возможностей, предоставляемых ему приложением (функций приложения):
создать новый заказ;
задать атрибуты поиска заказа;
найти заказ по текущим атрибутам поиска;
открыть текущий заказ на редактирование;
открыть список клиентов для добавления в текущий заказ;
открыть список товаров для добавления в текущий заказ;

просмотреть список клиентов;
выбрать клиента из списка клиентов;
добавить атрибуты текущего клиента к поиску заказа;
ввести данные нового клиента в текущий заказ;
редактировать данные текущего клиента в списке клиентов;
добавить текущего клиента в текущий заказ;
сохранить данные о текущем клиенте;
просмотреть список товаров;
задать атрибуты поиска товаров;
найти товар по текущим атрибутам;
выбрать товар из списка товаров;
просмотреть подробные данные текущего товара;
добавить атрибуты текущего товара к поиску заказа;
редактировать данные по текущему товару в текущем заказе;
добавить данные текущего товара в текущий заказ;
сохранить текущий заказ;
распечатать информацию по текущему заказу;
сформировать счет по текущему заказу.
Соответствие приведенных операций функциональным блокам, экранным формам и навигационным переходам указано на рисунке 1.
Далее, необходимо сгруппировать операции таким образом, чтобы их группы соответствовали пунктам главного меню. В рассматриваемом примере предлагается сформировать следующие группы.
1.Действия над объектами. В качестве объектов выступают заказ, клиент, товар (таблица 2).
Таблица 2 - Группа Действия

Действия	Объект	Примечания
Создать	Заказ Клиент	1 10
Открыть	Заказ Клиент Товар	4 11 18
Сохранить	Заказ Клиент	22 13
Выбрать (отобрать для добавления)	Клиент Товар Атрибуты клиента Атрибуты товара	12 (в заказ) 21 (в заказ) 9 (к поиску) 19 (к поиску)
Печать	Заказ	23
Счет	Заказ	24

2. Поиск. Специфическое действие, выделено отдельно; объекты – заказ (3), товар (16).
3. Работа со списками. Объекты – клиент, заказ (таблица 3).

Таблица 3 - Списки

Списки	Операции	Примечания
Клиенты	Просмотреть Открыть для выбора (добавления) в заказ	7 5
Товары	Просмотреть Открыть для выбора (добавления) в заказ	14 6

4. Стандартными являются такие разделы как Файл и Справка. Их тоже следует включить в главное меню приложения.
Составить граф состояния меню и провести проверку возможных переходов по графу в соответствии с пользовательскими сценариями.
Рассмотрим состояния меню для приведенного примера. Для простоты не будем учитывать состояния меню, связанные доступностью стандартных разделов Файл и Справка и их подразделов. Различные состояния прототипа меню можно представить таблицами 4-11. Разделы меню и команды, недоступные в данном состоянии выделены серым цветом. Для доступных команд в скобках указаны номера соответствующих операций.

Таблица 4 – Журнал заказов (состояние M1)

Действия	Поиск	Списки
Создать (1) Открыть (4) Сохранить Выбрать Печать Счет	Найти (3)	Клиенты (7) Товары (14)

Таблица 5 – Текущий заказ (состояние M2)

Действия	Поиск	Списки
Создать Открыть Сохранить (22) Выбрать Печать (23) Счет (24)		Клиенты (5) Товары (6)

Таблица 6 – Список клиентов (состояние M3, переход по команде 7)

Действия	Поиск	Списки
Создать (10) Открыть (11) Сохранить Выбрать (9) Печать Счет		

Таблица 7 – Список клиентов (состояние M4, переход по команде 5)

Действия	Поиск	Списки
Создать (10) Открыть (11) Сохранить Выбрать (12) Печать Счет		

Таблица 8 – Карта клиента (состояние M5)

Действия	Поиск	Списки
Создать Открыть Сохранить		

Выбрать (13) Печать Счет		

Таблица 9 – Список товаров (состояние M6, переход по команде 14)

Действия	Поиск	Списки
Создать Открыть (18) Сохранить Выбрать (19) Печать Счет	Найти (16)	

Таблица 10 – Список товаров (состояние M7, переход по команде 6)

Действия	Поиск	Списки
Создать Открыть (18) Сохранить Выбрать (21) Печать Счет	Найти (16)	

Таблица 11 – Карта товара (состояние M8)

Действия	Поиск	Списки

Граф состояний меню можно представить следующим образом (рисунок 2).

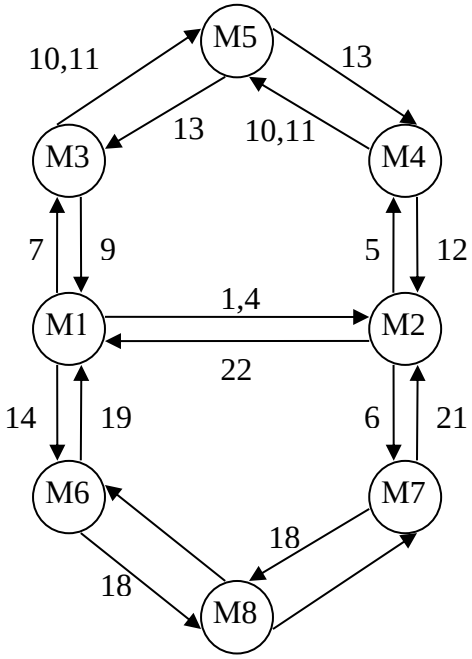


Рисунок 2 – Граф состояний меню

Следует подчеркнуть, что прототип меню в данном примере создается только под одну определенную категорию пользователей. Кроме того, в примере не предусмотрено развитие программного продукта, следовательно, отсутствует расширяемость функций.

№6. Низкоуровневое проектирование интерфейса: количественная оценка и построение прототипа

1 Цели работы
1.1 Закрепить теоретические знания по разработке пользовательского интерфейса.
1.2 Развить навыки создания вариантов прототипов интерфейса пользователя.
1.3 Получить практические навыки по количественной оценке интерфейса на этапе низкоуровневого проектирования.
1.4 Закрепить принципы обоснования выбора прототипа интерфейса по его количественной оценке.

2 Задание
2.1 Используя разработанное приложение, провести количественную оценку элементов интерфейса. Метод количественной оценки – GOMS, информационная производительность, символьная эффективность.
2.2 По результатам количественной оценки сделать выводы о возможности усовершенствования интерфейса.
2.3 При возможности внести необходимые усовершенствования в модели форм и реализовать их в среде разработки приложения. Каждую форму следует снабдить описанием навигации по ней. После улучшения получить те же количественные оценки для усовершенствованного варианта.
2.4 Составить отчет со скриншотами, пояснениями и выводом.

3 Краткие теоретические сведения
Для анализа качества интерфейсов используется множество количественных и эвристических методов. Одним из лучших подходов к количественному анализу моделей интерфейсов является классическая модель GOMS (goals, objects, methods and selection rules).
Метод, использующий модель GOMS, основан на разбиении всех действий пользователя на отдельные составляющие. Для каждой из них с помощью тщательных лабораторных исследований получен набор временных интервалов, необходимых для ее выполнения. В таблице 1 приведена номенклатура элементарных действий и соответствующие временные интервалы.
Таблица 1

Нажатие клавиши клавиатуры, включая клавиши Alt, Ctrl, Shift	0,28 с	К
Нажатие клавиши мыши	0,1 с	М
Указание – перемещение курсора мыши, чтобы указать какую-либо позицию на экране монитора	1,1 с	П
Перемещение- перенос руки пользователя с клавиатуры на мышь или обратно	0,4 с	В
Ментальная подготовка – мысленный выбор пользователем своего следующего элементарного действия	1,2	Д
Ответ – реакция системы на элементарное действие пользователя	-	Р

Широкая изменяемость каждой из представленных мер объясняет, почему эта модель не может использоваться для получения абсолютных временных значений с высокой степенью точности. Но этот метод вполне пригоден для проведения сравнительной оценки

между какими-либо двумя моделями интерфейса по уровню эффективности их использования.

Расчет времени, необходимого для выполнения некоторого действия начинают с разбиения его на элементарные действия, которые соответствуют номенклатуре приведенной в таблице 1. Проще всего выделить движения К, М, П, В. Проблему составляет определение моментов, когда пользователь должен остановиться, чтобы выполнить бессознательную ментальную операцию. Рассмотрим основные правила по выявлению этих моментов (таблица 2).

Таблица 2

Правило 0 Начальная расстановка операторов Д	Операторы Д следует устанавливать перед всеми операторами К и М (нажатие клавиши), также перед всеми операторами П, предназначенными для выбора команд. Но перед операторами П, предназначенными для указания на аргументы этих команд, ставить оператор Д не следует.
Правило 1 Удаление ожидаемых операторов Д	Если оператор, следующий за оператором Д, является полностью ожидаемым с точки зрения оператора, предшествующего Д, то этот оператор Д может быть удален. Если пользователь перемещает мышь с намерением нажать на ее кнопку по достижении цели движения, то в соответствии с этим правилом следует удалить оператор Д, установленный по правилу 0. Так последовательность действий П Д К преобразуется в П К.
Правило 2 Удаление операторов Д внутри когнитивных единиц	Если строка Д К Д К Д К… принадлежит когнитивной единице, то следует удалить все операторы Д, кроме первого. Когнитивной единицей является непрерывная последовательность вводимых символов, которые образуют название команды или аргумент. Например *Y, перемещать, 4564.23* – это когнитивные единицы.
Правило 3 Удаление операторов Д перед последовательными разделителями	Если оператор К означает лишний разделитель, стоящий в конце когнитивной единицы (например, разделитель команды, следующий сразу за разделителем аргумента этой команды), то следует удалить оператор Д , стоящий перед ним.
Правило 4 Удаление операторов Д, которые являются прерывателями команд	Если оператор К является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор Д, стоящий перед ним. (Добавление разделителя станет привычным действием, и поэтому разделитель станет частью строки и не будет требовать специального оператора Д.) Но если оператор К является разделителем строки аргументов или любой другой изменяемой строки, то оператор Д следует сохранить перед ним.
Правило 5 Удаление перекрывающих операторов Д	Любую часть оператора Д, которая перекрывает оператор Р, означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.

В этих правилах под ***строкой*** понимается некоторая последовательность символов. ***Разделителем*** считается символ, которым обозначено начало или конец значимого фрагмента текста, такого как, например, слово естественного языка или телефонный номер. Так, пробел является разделителем для большинства слов, а точка используется в конце предложений для разделения. В качестве разделителей могут выступать скобки для ограничений пояснений или замечаний и т.д.

Если для выполнения команды требуется дополнительная информация, она называется здесь ***аргументом*** данной команды.
Правила GOMS позволяют определит время, необходимое пользователю для выполнения любой . четко сформулированной задачи, для которой данный интерфейс предусмотрен. Однако этого метода недостаточно, чтобы оценить насколько быстро должен работать интерфейс – его производительность. Рассмотрим подробнее критерий эффективности Раскина, который оценивает информационную производительность интерфейса.
Информационная производительность интерфейса E определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь. Параметр E изменяется в пределах [0, 1]. В параметре E учитывается только информация, необходимая для задачи, и информация вводимая пользователем. Несколько методов действия могут иметь одинаковую производительность E, но иметь разное время выполнения. Возможно, что один метод имеет более высокий показатель E, но действует медленнее, чем другой метод.
Информация измеряется в битах. Один бит представляет собой один из двух альтернативных вариантов (0 или 1; да или нет) и является единицей информации. При количестве n равновероятных вариантов суммарное количество передаваемой информации определяется как

$$\log_2 n$$

Количество информации для каждого варианта определяется как

$$(1/n)\log_2 n \qquad (1)$$

Если вероятности для каждой альтернативы не являются равными и i-я альтернатива имеет вероятность $p(i)$, то информация, передаваемая этой альтернативой определяется как

$$p(i)\log_2(1/p(i)) \qquad (2)$$

Общее количество информации является суммой по всем вариантам выражений (1) или (2)
При использовании мыши в качестве устройства ввода информации, количество информации оценивают подобным же образом. Если экран поделен на две равные области: одна соответствует «да», а вторая соответствует «нет», то щелчок мыши в одной из этих областей будет передавать 1 бит информации. Если имеется n равновероятных объектов, то нажатием на один из них сообщается $\log_2 n$ бит информации. Если объекты имеют разные вероятности, используется сумма значений количества информации, полученные по формуле (2).
При передаче информации нажатием клавиши ее количество зависит от общего числа клавиш и относительной частоты использования каждой из них. То есть нажатия клавиш могут использоваться как приблизительная мера информации. Если на клавиатуре имеется 128 клавиш, и каждая из них используется с одинаковой частотой, то нажатие любой из них будет передавать

$$\frac{1}{128}\log_2(128) = 7 \text{ бит.}$$

В действительности частота использования клавиш существенно изменяется (например, клавиша ***е*** в русской раскладке клавиатуры используется наиболее часто) с учетом этого можно считать, что каждое нажатие клавиши передает ≈5 бит информации.
Иногда на практике удобнее использовать символьную эффективность вместо информационной производительности. Она определяется как минимальное количество символов, необходимое для выполнения задачи, отнесенное к количеству символов, которое в данной модели интерфейса требуется ввести пользователю.

3 Порядок выполнения работы

Рассмотрим пример с разработкой пользовательского интерфейса приложения создаваемого для мелкой фирмы (лабораторная работа №1). Программное обеспечение предполагает наличие справочников:
1) со списком клиентов фирмы, причем пользователь – в рассматриваемом примере это представитель обслуживающего персонала – может вносить в него изменения и добавлять новых клиентов;
2) со списком товаров, реализуемых фирмой. Пользователь не может вносить в него изменений.
Кроме того, создается список оформленных заказов – журнал заказов. Предположим, что такой список включает заказы за определенный период, например месяц. На следующий месяц создается новый журнал. Пользователь может вносить изменения в имеющиеся заказы, а также оформлять новые заказы.
Основные экранные формы известны из результата выполнения лабораторной работы №1:
Журнал заказов;
Текущий заказ;
Список клиентов;
Карта клиента;
Список товаров;
Карта товара.
Работа пользователя начинается с первой из них. Эта форма должна обеспечивать выполнение следующих операций:
создать новый заказ;
задать атрибуты поиска заказа;
найти заказ по текущим атрибутам поиска;
открыть текущий заказ на редактирование;
просмотреть список клиентов;
просмотреть список товаров;
Все эти операции пользователь может выполнить с помощью команд, собранных в разделах главного меню.
Выполним на примере первой из операций, связанной с созданием нового заказа, расчет времени по правилам GOMS.
При использовании команды меню эта операция раскладывается на следующие действия:
перемещение руки к мыши, В;
указание на раздел меню **Действия**, П;
нажатие клавиши мыши, М;
указание на команду **Создать**, П;
нажатие клавиши мыши, М.
В соответствии с правилом 0 расстановки ментальных операторов Д получим следующую последовательность операторов
В Д П Д М Д П Д М
В соответствии с правилом 1 следует удалить ожидаемые операторы Д
В Д П М Д П М
Остальные правила в этом примере не используются. Складывая соответствующие значения операторов получим общее время
0,4+1,2+1,1+0,1+1,2+1,1+0,1=5,2 с.
Очевидно, что это время велико. Необходимо предусмотреть другие варианты выполнения команды. Например, использование «горячих» клавиш или использование командной кнопки, запускающей команду на выполнение. Произведем оценку этих вариантов.
При использовании командной кнопки операция по созданию нового заказа раскладывается на действия:
перемещение руки к мыши, В;

указание на командную кнопку **Создать**, П;
нажатие клавиши мыши, М.
При расстановке операторов Д здесь также используются правила 0 и 1. Результирующая последовательность выглядит как
В Д П М
Общее время на выполнение операции составляет
0,4+1,2+1,1+0,1=2,8 с.
Это почти в 2 раза быстрее, чем предыдущий вариант выполнения команды.
При использовании «горячих» клавиш, например традиционного для создания нового объекта сочетания клавиш Ctrl+N, операция состоит из следующих действий:
перемещение руки к клавиатуре, В;
нажатие клавиши Ctrl, К;
нажатие клавиши N, К.
Согласно правилу 0 получаем последовательность операторов
В Д К Д К
Правило 1 в данном случае не работает, зато используется правило 2, согласно которому удаляются операторы Д внутри когнитивных единиц, каковой является комбинация клавиш Ctrl+N. Результирующая последовательность представлена как
В Д К К
Это соответствует времени
0,4+1,2+2·0,28=2,16 с.
Последний вариант самый быстрый. Однако рассматриваемая категория пользователей обладает низкой мотивацией к обучению, это означает, что пользователи могут не знать нужного сочетания клавиш, и они не стремятся ни узнать, ни запомнить его. Поэтому в интерфейсе желательно предусмотреть оба «быстрых» варианта выполнения команды.
Рассмотрим другую операцию, связанную с открытием текущего заказа. Очевидно, что она во многом схожа с предыдущей, однако требует задания аргумента – записи в журнале заказов, которая будет признана текущей. Для перемещения по журналу можно использовать клавиши клавиатуры **Home**, ↑, ↓, **End** или традиционные команды (командные кнопки) навигации по базе данных:
перейти к первой записи;
перейти к предыдущей записи;
перейти к следующей записи;
перейти к последней записи.
Рассмотрим оба этих варианта.
При использовании клавиш клавиатуры имеем:
перемещение руки к клавиатуре, В;
нажатие клавиши **Home**, К;
нажатие n раз клавиши ↓,К.
Результирующая последовательность операторов
В Д К Д К Д К Д К…
Нажатия клавиши не составляют единой когнитивной единицы, операторы Д исключить нельзя. После каждого нажатия пользователь принимает решение, следует ли ему нажать на клавишу повторно. Расчет по времени
0,4+1,2+0,28+n·(1,2+0,28)=1,88+n·1,68 с.
При использовании командных кнопок навигации получим:
перемещение руки к мыши, В;
нажатие кнопки **Перейти к первой записи**, М;
нажатие n раз кнопки **Перейти к следующей записи**, М.
Результирующая последовательность составляется аналогичным образом
В Д П М Д П М Д М Д М…

Расчет по времени
0,4+1,2+1,1+0,1+1,2+1,1+0,1+(n-1)·(1,2+0,1)=5,2+(n-1)·1,3 с
В зависимости от расположения нужной записи в журнале (значения n) предпочтительным может оказаться тот или иной вариант.
Гораздо проще для пользователя вариант с использованием техники прямого манипулирования, с использованием скроллинга - прокрутки. Рассмотрим, какие шаги должен выполнить пользователь в этом случае:
перемещение руки к мыши, В;
указание на область прокрутки, П;
нажатие клавиши мыши, М;
прокручивание скроллинга, С;
указание на нужную запись списка, П;
нажатие на клавиатуру мыши, М.
Здесь необходимо дать некоторые пояснения. При выполнении шага (3), осуществляется нажатие на клавишу мыши и ее удерживание. Время, поставленное в соответствие оператору М, учитывает как нажатие, так и отпускание клавиши, поэтому шаг (3) реально выполняется за 0,05 с (М/2). Но тогда после выполнения шага (4), необходимо учесть время, требуемое для отпускания клавиши мыши, что составляет 0,05 с (М/2). В приведенной выше последовательности действий, нажатие и отпускание мыши объединено в один шаг. Такое объединение становится возможным, потому что, согласно правилу 1 в последовательности
(М/2) С (М/2)
операторы Д не вставляются.
Время необходимое на прокручивание скроллинга можно оценить только экспериментальным путем. Приближенно можно считать, что для его выполнения необходимо 3 с.
Результирующая последовательность операторов выглядит как
В Д П М С Д П М
Это соответствует оценке времени
0,4+1,2+1,1+0,1+3+1,2+1,1+0,1=8,2 с.
При «удачном» расположении записи в журнале, пользователю вообще не придется работать с прокруткой. Последовательность операторов его действий будет выглядеть как
В Д П М
что составляет по времени 2,8 с.
Вариант с использованием прокрутки можно считать более предпочтительным. Но учитывая низкий уровень владения компьютером и минимальный предварительный тренинг, стоит предусмотреть также вариант с использованием клавиш ↑, ↓.
Аналогичные рассуждения можно привести относительно других операций.
Примерный вид экранной формы **Журнал заказов** приведен на рисунке 1.

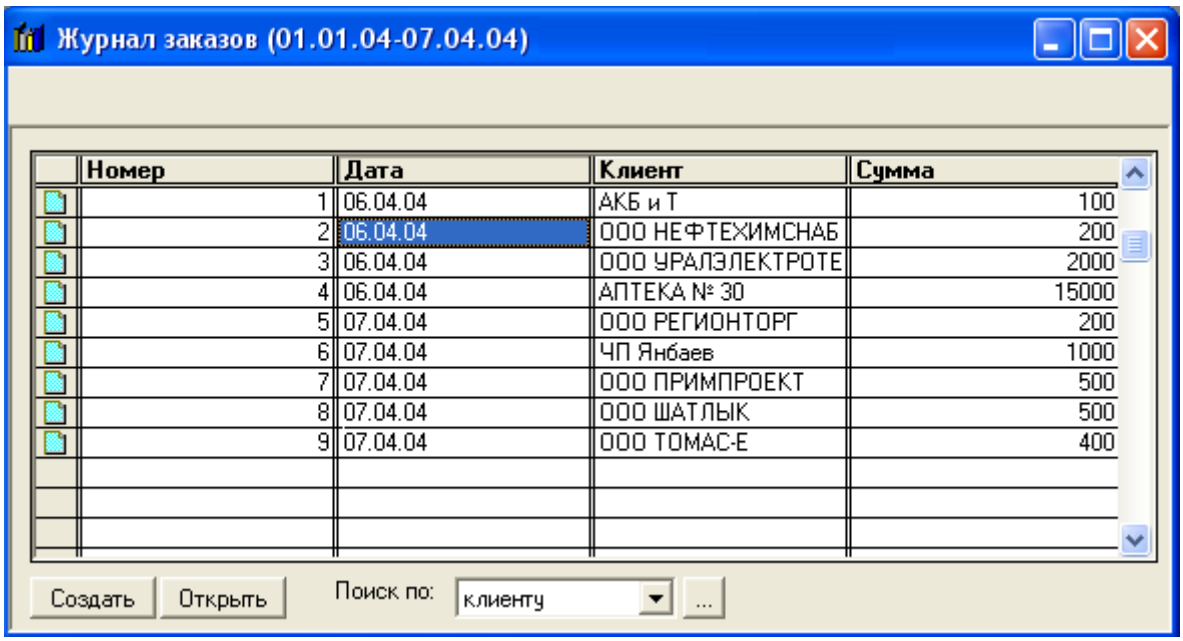


Рисунок 1 – Экранная форма **Журнал заказов**

Рассмотрим, каким образом дается оценка производительности элементов интерфейса данной формы. В качестве примера выберем предложенную реализацию для операции поиска по данным справочников. Предусмотрено два варианта справочников: клиентов и товаров. Причем, для задания атрибутов поиска предполагается выполнять переход к формам этих справочников. В начальном прототипе формы переход осуществлялся путем нажатия на командную кнопку **Просмотреть** или командную кнопку **Найти**. Будем считать равновероятным выбор одного из двух справочников; вероятность использования кнопки **Просмотреть** оценим в 30% (p=0,3), кнопки **Найти** – 70% (p=0,7). Вероятности различных вариантов составят:

выбор клиента по кнопке **Просмотреть** – 0,5·0,3=0,15;

выбор товара по кнопке **Просмотреть** – 0,5·0,3=0,15;

выбор клиента по кнопке **Найти**– 0,5·0,7=0,35;

выбор товара по кнопке **Найти**– 0,5·0,7=0,35.

Информационное содержание рассматриваемого фрагмента интерфейса как

$$0{,}15\cdot\log_2\left(\frac{1}{0{,}15}\right)+0{,}15\cdot\log_2\left(\frac{1}{0{,}15}\right)+0{,}35\cdot\log_2\left(\frac{1}{0{,}35}\right)+0{,}35\cdot\log_2\left(\frac{1}{0{,}35}\right)=0{,}41+0{,}41+0{,}53+0{,}53=1{,}88$$

Теоретически, если пользователь решил выполнить функцию поиска, ему минимально необходимо только определить каким из двух справочников он желает воспользоваться. Следовательно, минимальное количество информации необходимое ему для решения этой задачи определяется как

$$\log_2 2=1$$

Информационная производительность

$$E=\frac{1}{1{,}88}=0{,}53$$

Повысить информационную производительность можно, для этого следует предусмотреть объединить обе команды или осуществлять переход к соответствующей экранной форме непосредственно при выборе пользователем нужного ему справочника (рисунок 1).

Оценить символьную эффективность можно только для тех элементов интерфейса, в которых осуществляется ввод символьной информации. Этот метод количественного анализа не представляет особых сложностей.

№7. Проектирование справочной системы

Цель работы: изучение методики создания файлов справочной системы Windows (*.hlp) при разработке приложений.

Задание:

1. Создать глоссарий – перечень уникальных понятий.
2. Спроектировать справочную систему для приложения.
3. Обеспечить возможность поиска по ключевым словам.
4. Решить такие вопросы проектирования справочной системы как вид основного окна, введение изображений, наличие горячих областей и ссылок.
5. В результате выполнения работы получить программу с функциональным справочником, поиском, указателем. Возможно также и создание контекстной справки.
6. Составить отчет со скриншотами, пояснениями и выводом.

Краткие теоретические сведения.
Разработка справочной системы состоит из двух основных этапов:
Создание файлов или нескольких файлов, содержащих темы справок, например, с помощью Microsoft Word.
Компиляция справки в файл и отладка всей справочной системы, с помощью специальных программ, например, HCRTF - Microsoft Help Workshop.
При создании справочной системы необходимо в первую очередь продумать систему в целом. А именно решить следующие вопросы:
об информации, выносимой в основное, дополнительное, всплывающие окна;
о включении разделов в предметный указатель и окно содержания;
о представлении информации в основном окне (например, наличие начальной части, неподдающейся прокрутке) и общем стиле справки (наличие кнопок, пиктограмм, "горячих" областей);
о разделении функций между основным и дополнительными окнами.
При создании файлов тем справок можно использовать текстовый редактор Microsoft Word, созданный файл должен сохраняться в формате RTF.
Каждая тема (кадр) должна начинаться с новой страницы (разрыв страницы выполняется нажатием клавиш Ctrl-Enter). Первой должна располагаться страница Содержание, порядок остальных безразличен.
При написании темы можно использовать многие возможности оформления шрифта. Но если нет уверенности, что соответствующие шрифты будут иметься на компьютерах потенциальных пользователей, лучше использовать обычные системные шрифты (MS Sans Serif).
По умолчанию при отображении текста в окне справки часть строки, которая не помещается в слишком узком для нее пространстве, переносится на новую строку. В случаях, когда это нежелательно, следует выделить соответствующие строки, и выполнить команду Формат | Абзац и в открывшемся окне на странице Положение на странице установить опцию Не разрывать абзац. Тогда, если ширины справки не хватает , чтобы отобразить всю длину отмеченных таким образом строк, в окне появится полоса горизонтальной прокрутки.
Для того чтобы при перемещении по тексту темы какая-то его часть (заголовок, начальная часть) не прокручивалась вертикальной прокруткой и всегда оставалась на экране, следует выделить соответствующий текст, выполнить команду Формат | Абзац и в открывшемся

окне на странице Положение на странице установить опцию Не отрывать от следующего.
В кадр могут специальным образом включаться рисунки, кнопки и т.д.
Добавлять изображения в тему можно, например, в виде файлов .bmp, пользуясь или буфером обмена, или одной из команд:
{bmc <имя файла> }- размещение рисунка как обычного символа по ходу текста;
{bml <имя файла>} – размещение рисунка с левой стороны;
{bmr <имя файла> } – размещение рисунка с правой стороны.
Темы могут содержать горячие области: выделенные слова или кнопки, позволяющие пользователю выполнять переход от одной темы к другой. Кнопка вставляется командой {button <надпись>, <список макросов>}. В этой команде указывается текст надписи, которая будет присутствовать на изображаемой кнопке, а также перечень макросов, выполняемых при ее нажатии. Рассмотрение всего множества макросов, которые можно использовать при проектировании справочной системы выходит за рамки данной лабораторной работы. Описание этих макросов можно получить в файле hcw.hlp .
Каждая тема снабжается сносками (команда Вставка|Сноска), для которых используются определенные символы. Все сноски располагаются в первых позициях кадра. Символы, используемые для сносок различного типа приведены в таблице 1, там же раскрыто и назначение сносок каждого типа.
Таблица 1

Символ сноски	Назначение сноски
#	Обозначает уникальный идентификатор темы. По нему на данную тему могут ссылаться другие темы. Этому идентификатору ставится в соответствие номер, по которому на данную тему ссылается использующее справку приложение.
K	Позволяет отображать тематику кадра в предметном указателе. Название темы в предметном указателе представляет собой текст сноски K. Для одного кадра можно ввести несколько обозначений, разделяемых точкой с запятой. Пример KОбъект Поле; Объекты В предметном указателе две строки «Объект Поле» и «Объекты» будут вызывать один и тот же кадр. Можно сформировать двух уровневые ссылки: темы первого и второго уровня разделяются запятой. Пример KОбъект Поле, Создание; Объект Поле, Ввод данных; Объект Поле, Модификация свойств Кроме того, элементы, указанные в сносках K, используются при организации переходов между темами переход по ключевым словам (с помощью макроса Klink)
$	Определяет заголовок данной темы, используется в некоторых режимах работы, например, Поиск, Назад и др. Например, если указанное пользователем ключевое слово соответствует нескольким темам, то от пользователя требуется уточнение. В этом случае появляется окно Найденные разделы. В нем отображается текст сносок $. Поэтому сноски $ включают только в те кадры, которые имеют в своих сносках K элементы Klink, используемые в сносках K других кадров.
A	Используется для организации перехода по ключевым словам (с помощью макроса Alink)
+	Используется для указания последовательности просмотра тем. Включаются в кадр только тогда, когда соответствующие кнопки (кнопки Browse) предусмотрены в окне справки. Если сноски + включены в кадр, но их значения не указаны, последовательность просмотра тем будет установлена

	автоматически в соответствие с последовательностью тем в файле. Тексты сносок могут быть номерами или идентификаторами.
!	Используется для указания макросов, которые должны сработать перед появлением окна темы (для сложных справочных систем).
*	Используется для указания связанных друг с другом тем, которые должны быть встроены в справку (при создании справочных систем связанных программных продуктов).
>	Используется для указания идентификатора окна, в котором должна отображаться тема

В темы можно вводить переходы от одной темы к другой. Существуют непосредственные переходы и переходы по ключевым словам.

Непосредственный переход выполняется при щелчке пользователя на горячей области текста. Для этого сразу после нужных слов (без пробела) надо написать идентификатор темы, на которую нужно перейти. Соответствующие слова следует выделить двойным подчеркиванием (команда Формат|Шрифт, диалоговое окно Шрифт, опция Подчеркивание в положении Двойное), а идентификатор темы оформить как скрытый текст (команда Формат|Шрифт, диалоговое окно Шрифт, раздел Эффекты, опция Скрытый установлена, опция Подчеркивание в положении "(нет)").

Можно отображать тему, на которую осуществляется переход во всплывающем окне. Такие окна обычно используются для дополнительной информации и различных пояснений. Всплывающее окно не удаляет окно темы его вызвавшей, но при любом действии пользователя исчезает. Переход к теме, отображаемой во всплывающем окне, осуществляется аналогично, но с выделением одинарным подчеркиванием.

Переходы по ключевым словам позволяют осуществлять два макроса, имеющие одинаковый синтаксис Klink и Alink. Например,

Klink ("<список ключевых слов>", <тип>, "<идентификатор темы>",
<имя окна>)

Обязательным элементом вызова макроса является только первый - "<список ключевых слов>". Он представляет собой одно или несколько ключевых слов или словосочетаний, перечисленных через точку с запятой. Если входящее в этот перечень словосочетание содержит запятую, то весь список заключается в двойные кавычки. Поиск ведется по первому слову, при нахождении нескольких тем пользователю предъявляется окно Найденные разделы, если не найдено ни одной темы, начинается поиск по второму ключевому слову и т.д.

<тип> определяет реакцию на найденные или не найденные ключевые слова и может принимать следующие значения (таблица 2).

Таблица 2

Символическое значение	Численное значение	Описание значения
JUMP	1	Если найдена только одна тема, соответствующая ключевым словам, то на нее сразу осуществляется переход
TITLE	2	Если ключевое слово находится более чем в одном файле справки (при справке, состоящей из нескольких файлов), то в окне Найденные разделы после названия темы пишется имя файла, определенное в файле содержания (*.cnt)
TEST	4	Возвращается величина указывающая нашлось или нет хотя бы одно соответствие ключевым словам.

<идентификатор темы> определяет, что если не найдено соответствия ключевым словам, то появляется всплывающее окно с текстом, содержащимся в теме, на которую указывает

данный идентификатор. Если идентификатор не задан, то при безуспешном поиске появляется окно с текстом «Дополнительные сведения отсутствуют».
<имя окна> задает окно для отображения. Если этот параметр не задан, то используется окно, заданное в кадре темы или окно по умолчанию.
В качестве примера использования макросов можно рассмотреть следующий текст
Объект Поле!Klink (Объект Поле; Объекты) позволяет осуществить ввод данных.
В готовом файле справки словосочетание «Объект Поле» будет выделено цветом. Щелкнув клавишей мыши на этом словосочетании, пользователь увидит список тем, содержащих в своих сносках K ключевые слова: «Объект Поле» и «Объекты». Если такая тема является единственной, то сразу осуществится переход на нее.
В тексте может быть использован оператор вида
{button Объекты, Alink (Объекты)}
Это приведет к появлению в кадре кнопки с надписью «Объекты», при нажатии на которую будет осуществляться поиск тем, у которых ключевое слово присутствует в сносках A.
Когда сформированы темы проектируемой справочной системы, разработана ее структура, решен вопрос об общем стиле справки, переходят ко второму этапу проектирования – компиляции справки в файл *.hlp .
Компиляция и отладка справочной системы может проводиться с использованием программы HCRTF - Microsoft Help Workshop, расположенной в каталоге Delphi…\Help\Tools. Эта программа позволяет создать файл Проекта справки, без которого ее нельзя компилировать, файл содержания справки, а также проверить справочную систему в работе.
Для создания файла проекта простой справочной системы нужно выполнить следующие действия:
Выполнить File|New и в открывшемся окне выбрать опцию Help Project.
В окне Project File Name задать имя и каталог файла Проекта справки (каталог выбрать тот, в котором лежит файл текстов тем .rtf).
В открывшемся окне Проекта нажать кнопку Files….
В появившемся окне Topic Files нажать кнопку Add… и выбрать среди файлов подготовленный файл текстов справки.
Нажать OK.
Когда файл проекта создан, необходимо сохранить и откомпилировать его – кнопка Save and Compile, использовать ее следует каждый раз при изменении файла Проекта. После компиляции можно просмотреть сведения о результатах компиляции, и если она завершилась удачно (указан размер файла) просмотреть файл справочной системы.
Просмотр и работа по отладке выполняется с помощью команды File|Run WinHelp.
Чтобы создать новый файл содержания необходимо выполнить команду File|New и в окне New выбрать опцию Help Contents, в результате чего появится окно с загруженным в него файлом содержания. В верхнем левом поле ввода задается имя файла *.hlp, верхнем правом поле ввода задается заголовок, который должен появиться в окне при работе со справочной системой. Нижнее поле ввода заполняется с помощью кнопок Add Above … и Add Below…. Каждая его строка соответствует либо заголовку (будет отображаться в виде закрытой книги), либо теме (отображается в виде листа с вопросом). Для заголовка задается только его текст, для темы задается название и идентификатор. Кнопки позволяют формировать многоуровневую структуру файла содержания. Кнопка Edit… позволяет редактировать выделенную строку, а кнопка Remove… удалять.
Файл содержания имеет расширение *.cnt.

№8. Диалог виды(проектирование диалога,диалог на основе экранных форм, выбор структуры диалога, диалог типа Вопрос – Ответ…)

Цель: работы: изучение методики создания диалогов либо другой интерактивной документации.

Назначение: овладение навыками и приемами создания пользовательского интерфейса в среде Windows. Эти же приемы будут применяться в последующем во время создания прикладных приложений и других дополнений к Windows. Навыки работы с интерфейсами позволят кастомизировать и упростить операции целевого пользователя.

Задания

1. Описать вид диалога, который используется в приложении, разработанном в предыдущих лабораторных работах.
2. Чем характеризуется выбор данного типа диалога?
3. Опишите другие варианты диалога для вашего приложения, какие элементы будут использоваться для организации того же функционала? Изобразить иной вариант диалога в виде эскиза.
4. Структура диалога типа вопрос-ответ?
5. Диалог на основе меню?
6. Диалог на основе экранного меню?
7. Составить отчет со скриншотами, пояснениями и выводом.

Краткая теория

Принцип «Обратной связи»

Необходимо обеспечивать обратную связь для действий пользователя. Каждое действие пользователя должно получать визуальное, а иногда и зву-ковое подтверждение того, что приложение восприняло введенную команду; при этом вид реакции, по возможности, должен учитывать природу выпол-ненного действия.
Обратная связь эффективна в том случае, если она реализуется свое-временно, т.е. как можно ближе к точке последнего взаимодействия пользо-вателя с системой. Когда компьютер обрабатывает поступившее задание, по-лезно предоставить пользователю информацию относительно состояния про-цесса, а также возможность прервать этот процесс в случае необходимости. Ничто так не смущает не очень опытного пользователя, как заблокированный экран, который никак не реагирует на его действия. Типичный начинающий пользователь способен вытерпеть только несколько секунд ожидания ответ-ной реакции от своего электронного «собеседника».

Интерфейс должен быть простым. При этом имеется в виду не упро-щенчество, а обеспечение легкости в его изучении и в использовании. Кроме того, он должен предоставлять доступ ко всему перечню функциональных возможностей, предусмотренных данным приложением. Реализация доступа к широким функциональным возможностям и обеспечение простоты работы противоречат друг другу

Проектирование диалога

Проводившиеся в свое время исследования психологических аспектов общения человека с компьютером показали, что следует всячески стремиться к дегуманизации этого общения, то есть пользователь не должен восприни-мать компьютер как полноценного собеседника. Тем не менее, обмен инфор-мацией между пользователем и компьютером (точнее, его программным обеспечением) по всем формальным признакам соответствует понятию «диа-лог» в общепринятом смысле. Можно перечислить основные правила, кото-рые должны соблюдаться, чтобы диалог оказался конструктивным:
– во-первых, участники диалога должны понимать язык друг друга;
– во-вторых, они не должны говорить одновременно;
– в-третьих, очередное высказывание должно учитывать как общий контекст диалога, так и последнюю информацию, полученную от собеседника.

Если собеседники обсуждают вопросы, относящиеся к какой-либо спе-циальной области, они должны придерживаться единой терминологии; если же один из них пытается что-то объяснить другому, ему следует сначала по-яснить основные термины и понятия.
К этому следует также добавить, что применение дополнительных вы-разительных средств способствует лучшему взаимопониманию. Иногда одна удачная иллюстрация заменяет десятки слов. Например, на вопрос "Как пройти в библиотеку?" лучше всего отвечать, имея под рукой карту города.
Таким образом, при проектировании пользовательского интерфейса необходимо определить:
 структуру диалога;
 возможный сценарий развития диалога;
 содержание управляющих сообщений и данных, которыми могут обмениваться человек и приложение (семантику сообщений);
 визуальные атрибуты отображаемой информации (синтаксис со-общений).

Выбор структуры диалога
Выбор структуры диалога — это первый из этапов, который должен быть выполнен при разработке интерфейса.
Рассмотренные ниже четыре варианта структуры диалога являются разновидностями структуры типа «вопрос — ответ», тем не менее каждая из них имеет свои особенности и наиболее удобна для определенного класса задач.

Диалог типа «Вопрос - Ответ»
Структура диалога типа «вопрос-ответ» основана на аналогии с обычным интервью. Система берет на себя роль интервьюера и получает информацию от пользователя в виде ответов на вопросы. Это наиболее известная структура диалога; все диалоги, управляемые компьютером, в той или иной степени состоят из вопросов, на которые пользователь отвечает. Однако в данной структуре этот процесс выражен явно. В каждой точке диалога система выводит в качестве подсказки один вопрос, на который пользователь дает один ответ. В зависимости от полученного ответа система может решить, какой следующий вопрос задавать. Структура предоставляет естественный механизм ввода как управляющих сообщений (команд), так и данных. Никаких ограничений на диапазон или тип входных данных, которые могут обрабатываться, не накладывается. Существуют системы, ответы в которых даются на естественном языке, но чаще используются предложения из одного слова с ограниченной грамматикой.

Диалог в виде вопросов и ответов в достаточной степени обеспечивает поддержку пользователя, так как даже краткий наводящий вопрос при разум-ном построении может быть самопоясняющим. Эта структура не гарантирует минимального объема ввода, оцениваемого по количеству нажатий клавиш, однако при подходящем подборе сокращений можно уменьшить любую из-быточность. Вместе с тем, структура обладает одним существенным недо-статком. Даже если ввод происходит достаточно быстро, для человека, кото-рый уже знает, какие вопросы задает система и какие ответы нужно на них давать, отвечать на всю серию вопросов довольно утомительно.

Диалог на основе меню

Меню является, пожалуй, наиболее популярным вариантом организа-ции запросов на ввод данных во время диалога, управляемого компьютером.

Существует несколько основных форматов представления меню на экране:

• список объектов, выбираемых прямым указанием, либо указанием номера (или мнемонического кода);
• меню в виде блока данных;
• меню в виде строки данных;
• меню в виде пиктограмм.

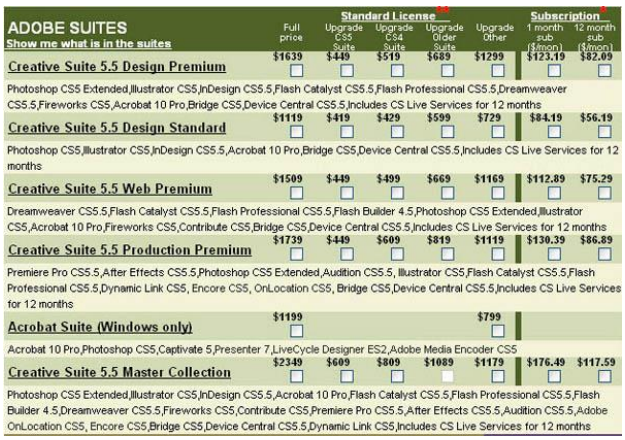
Меню в виде строки данных может появляться вверху или внизу экрана и часто остается в этой позиции на протяжении всего диалога. Таким обра-зом, посредством меню удобно отображать возможные варианты данных для ввода, доступных в любое время работы с системой. Если в системе имеется достаточно большое разнообразие вариантов действий, организуется иерар-хическая структура из соответствующих меню. Дополнительные *меню в виде блоков данных* «всплывают» на экране в позиции, определяемой текущим по-ложением указателя, либо «выпадают» непосредственно из строки меню верхнего уровня. Эти меню исчезают после выбора варианта.

Меню в виде пиктограмм представляет собой множество объектов вы-бора, разбросанных по всему экрану; часто объекты выбора заголовком во всплывающем окне или содержат графическое представление вариантов ра-боты.

Пользователь диалогового меню может выбрать нужный пункт, вводя текстовую строку, которая идентифицирует этот пункт, указывая на него непосредственно или просматривая список и выбирая из него. Система может выводить пункты меню последовательно, при этом пользователь выбирает нужный ему пункт нажатием клавиши.

Меню можно с равным успехом применять для ввода, как управляю-щих сообщений, так и данных. Приемлемая структура меню зависит от его размера и организации, от способа выбора пунктов меню и реальной потребности пользователя в поддержке со стороны меню. . Примеры:

Рис 1.1



ADOBE SUITES Show me what is in the suites	Standard License Full price	Upgrade CS5 Suite	Upgrade CS4 Suite	Upgrade Older Suite	Upgrade Other	Subscription 1 month sub ($/mon)	12 month sub ($/mon)
Creative Suite 5.5 Design Premium	$1639	$449	$519	$689	$1299	$123.19	$82.09
Photoshop CS5 Extended,Illustrator CS5,InDesign CS5.5,Flash Catalyst CS5.5,Flash Professional CS5.5,Dreamweaver CS5.5,Fireworks CS5,Acrobat 10 Pro,Bridge CS5,Device Central CS5.5,Includes CS Live Services for 12 months							
Creative Suite 5.5 Design Standard	$1119	$419	$429	$599	$729	$84.19	$56.19
Photoshop CS5,Illustrator CS5,InDesign CS5.5,Acrobat 10 Pro,Bridge CS5,Device Central CS5.5,Includes CS Live Services for 12 months							
Creative Suite 5.5 Web Premium	$1509	$449	$499	$669	$1169	$112.89	$75.29
Dreamweaver CS5.5,Flash Catalyst CS5.5,Flash Professional CS5.5,Flash Builder 4.5,Photoshop CS5 Extended,Illustrator CS5,Acrobat 10 Pro,Fireworks CS5,Contribute CS5,Bridge CS5,Device Central CS5.5,Includes CS Live Services for 12 months							
Creative Suite 5.5 Production Premium	$1739	$449	$609	$819	$1119	$130.39	$86.89
Premiere Pro CS5.5,After Effects CS5.5,Photoshop CS5 Extended,Audition CS5.5, Illustrator CS5,Flash Catalyst CS5.5,Flash Professional CS5.5,Dynamic Link CS5, Encore CS5, OnLocation CS5, Bridge CS5,Device Central CS5.5,Includes CS Live Services for 12 months							
Acrobat Suite (Windows only)	$1199				$799		
Acrobat 10 Pro,Photoshop CS5,Captivate 5,Presenter 7,LiveCycle Designer ES2,Adobe Media Encoder CS5							
Creative Suite 5.5 Master Collection	$2349	$609	$809	$1089	$1179	$176.49	$117.59
Photoshop CS5 Extended,Illustrator CS5,InDesign CS5.5,Acrobat 10 Pro,Flash Catalyst CS5.5,Flash Professional CS5.5,Flash Builder 4.5,Dreamweaver CS5.5,Fireworks CS5,Contribute CS5,Premiere Pro CS5.5,After Effects CS5.5,Audition CS5.5,Adobe OnLocation CS5, Encore CS5,Bridge CS5,Device Central CS5.5,Dynamic Link CS5,Includes CS Live Services for 12 months							

Рис 1.3

Структура типа меню - является наиболее естественным механизмом для работы с устройствами указания и выбора: меню представляет собой изображение тех объектов, которые выбираются пользователем. Если диалог состоит исключительно из меню, можно реализовать последовательный ин-терфейс, при котором пользователь применяет только устройства для указа-ния, однако такое постоянство редко достижимо на практике. Работа с этими устройствами не требует профессионального владения клавиатурой и для подготовленного пользователя это не самый быстрый способ выбора из ме-ню. Вместо указания пользователь может сообщить о своем выборе вводом соответствующего идентификатора.

Меню — это наиболее удобная структура диалога для неподготовлен-ных пользователей; жесткая очередность открытия и иерархическая вложен-ность меню может вызывать раздражение профессионала, замедлять его ра-боту. Традиционная структура меню недостаточно гибка и не в полной мере согласуется с методами адаптации диалога, такими, например, как опережа-ющий ввод, с помощью которого можно ускорить темп работы подготовлен-ного пользователя.
Пример:

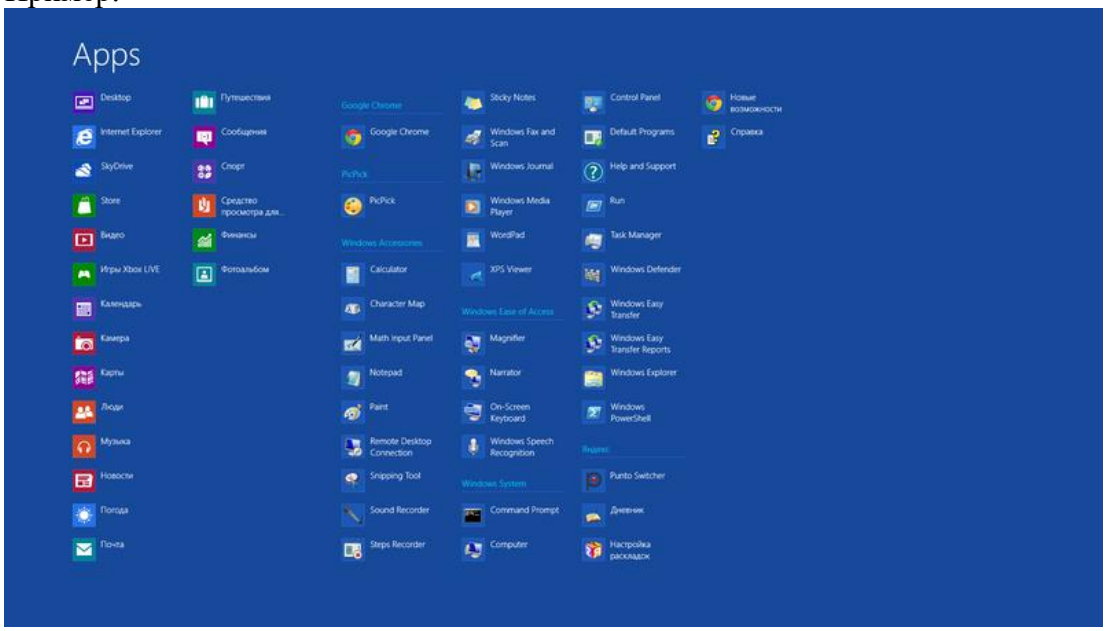


Рис 1.2Пример меню.

Диалог на основе экранных форм

Как структура типа «вопрос — ответ», так и структура типа меню предполагают обработку на каждом шаге диалога единственного ответа. *Диа-лог на основе экранных форм* допускает обработку на одном шаге диалога нескольких ответов. На практике формы используются в основном там, где учет какой-либо деятельности требует ввода достаточно стандартного набора данных. Человек, заполняющий форму, может выбирать последовательность ответов, временно пропускать некоторый вопрос, возвращаться назад для коррекции предыдущего ответа и даже «порвать бланк» и начать заполнять новый. Он работает с формой до тех пор, пока не заполнит ее полностью и не передаст системе. Программная система может проверять каждый ответ непосредственно после ввода или выждать и вывести список ошибок только после заполнения формы целиком. В некоторых системах информация, вво-димая пользователем, становится доступной только после нажатия клавиши «ввод» по окончании заполнения формы.

№9. Адаптация интерфейса для различных платформ.

Цель: работы: изучение методики адаптации интерфейса для различных платформ.

Назначение: овладение навыками и приемами создания пользовательского интерфейса в различных мобильных средах. Адаптация UX и UI приложения для другой платформы, сохранив целостность дизайна продукта. Навыки работы с мобильными платформами позволят оптимизировать операции целевого пользователя.

Задания
Написать программу для мобильного устройства (смартфона):
Выбрать среду разработки;
Определить функционал приложения;
Адаптировать интерфейс приложения для различного разрешения экрана.

Краткая теория

Существует три варианта мульти-платформенной адаптации UI:

1. Полное сохранение целостности бренда на всех платформах.
2. Соответствие нормам, которые приняты для определенной платформы.
3. Поиск баланса между перечисленными выше способами.

Подход 1: Ориентация на целостность бренда

Сохранение одного дизайна для всех платформ, игнорируя всяческие «родительские указания», — самый быстрый, простой и экономный подход, когда дело касается дизайна UI. Однако с программной частью это, к сожалению, не сработает: нестандартный пользовательский интерфейс сложнее в реализации, его разработка обойдется дороже, нежели использование обычных компонентов.

Унификация дизайна — это опасный подход. Подобные приложения называют подростками. Они не следуют стандартным правилам и часто сильно раздражают. Они считают себя не такими как все, особенными, однако на самом деле фактически ничем не отличаются от ровесников. Хотя бывают и приятные исключения.

Адаптируя интерфейс для другой платформы, необходимо себя поставить на место пользователя — совсем неважно, какой подход вы избрали. Способность создавать хороший интуитивный UX, сохраняя при этом целостность бренда, — качество, которое присуще только самым лучшим дизайнерам.

Подход 2: Ориентация на платформу

Ориентация на определенную платформу, в отличие от ориентации на целостность бренда, требует намного больших затрат времени и денег (если говорить только о дизайне интерфейса). При переносе дизайна довольно многие UX- и UI-элементы должны быть созданы с нуля для того, чтобы соответствовать всем нормам целевой платформы.

В данном случае фокус смещается с целостности бренда на стандартный UX, поскольку у пользователей есть свойство «привыкать» к платформе. В этом случае интуитивное восприятие приложения через хорошо знакомый дизайн может дать много преимуществ.

Любая мобильная программа должна быть максимально простой в использовании, даже если у нее очень много функций. В случае если у приложения много контента и богатая функциональность, то нужно ориентироваться на платформу.

Следование нормам платформы, а также предоставление пользователям опыта, к которому они привыкли, дает возможность оправдать их ожидания.

Тем не менее, где одни приложения имеют успех, других вполне может ожидать провал. Ориентация на платформу — это не лучшее решение для компании, которая хочет поддерживать уникальность своего бренда. В конце концов, бренд — лицо, помогающее приносить деньги. Поэтому не надо превращать свою программу в «примерного ученика», который всегда следует правилам, но при этом не очень популярен среди своих одноклассников.

Подход 3: Смешанный

Проявив определенную находчивость, можно остаться верным стандартам платформы, при этом сохраняя собственное лицо. Такой смешанный подход к мульти-платформенной адаптации дизайна — это очень сбалансированный синтез двух описанных выше подходов. И этот подход наиболее сложный.

Следуя смешанному подходу, нужно учитывать два вида пользователей: те, кто уже знаком с вашим продуктом, и те, кто им никогда не пользовался. Первые верны бренду, а вторые привыкли к особенностям платформы.

Дизайнеры, которые выбирают данный подход, в некотором смысле дипломаты, представляющие интересы бренда и одновременно налаживающие дружеские отношения с пользователями. Им необходимо определить, какие элементы интерфейса выделяют их продукт среди всех прочих, и найти специфические для определенной платформы решения, которые точно не повредят бренду.

Какой подход лучше

Даже вопреки тому, что смешанный метод кажется наилучшим, тем не менее, сегодня ни один из подходов, которые описаны, не является идеальным.

Выбор идентичности бренда как приоритета вполне может за собой повлечь целый ряд проблем с UX, даже если само приложение будет хорошо выглядеть. Ведь это пользователи должны решать, как именно должно выезжать меню, где сделать свайп, чтобы получить доступ к конкретной функции, и как вернуться на страничку профиля.

Приложения, которые ориентируются прежде всего на платформу, рискуют выглядеть очень стандартно, что отрицательно влияет на бренд. С иной стороны, подобные приложения все- равно могут иметь большой успех у широкой публики, в особенности если в них достаточно широкая функциональность.

Создавая дизайн современных приложений, следует помнить о том, что вы делаете это для реальных людей, которые будут пользоваться этими приложениями на вполне реальных устройствах и в реальном мире. На самом деле, важна не платформа, не бренд, и даже не ваша креативность.

Основная и дополнительная литература

Основная

1. Взаимодействие пользователей с интерфейсами информационных систем для мобильных устройств: исследование опыта [Электронный ресурс]: учебное пособие /Ткаченко О.Н. - М.: Магистр: ИНФРА-М, 2018. - 152 с. Режим доступа: http://znanium.com/catalog/product/937425

2. Разработка пользовательского интерфейса на основе системы Windows Presentation Foundation [Электронный ресурс]: учебник / А.В. Абрамян, М.Э. Абрамян; Южный федеральный университет. - Ростов-на-Дону; Таганрог: Издательство Южного федерального университета, 2017. - 301 с. Режим доступа: http://znanium.com/catalog/product/1020507

Дополнительная

3. Технология разработки программного обеспечения [Электронный ресурс]: учебное пособие / Л.Г.Гагарина, Е.В. Кокорева, Б.Д. Виснадул; Под ред. проф. Л.Г.Гагариной - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 400 с. Режим доступа: http://znanium.com/catalog.php?bookinfo=389963

4. Разработка пользовательского интерфейса [Электронный ресурс]: научно-практическое пособие / Т. Мандел; Пер. с англ. - М.: ДМК Пресс, 2007. - 416 с. Режим доступа: http://znanium.com/catalog.php?bookinfo=407684

5. Заботина Н.Н. Проектирование информационных систем [Электронный ресурс]: учебное пособие / Н.Н. Заботина. - М.: НИЦ ИНФРА-М, 2013. – 331с. Режим доступа: http://znanium.com/catalog.php?bookinfo=371912