МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФГБОУ ВО «ИНГУШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ГУМАНИТАРНО-ТЕХНИЧЕСКИЙ КОЛЛЕДЖ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «Визуализация интерфейса пользователя»

Специальность: 09.02.07 «Информационные системы и программирование»

/код, наименование /

Выполнил(а): студент(ка) 4 курса ИСП − 403

Хамхоев Зураб Вячеславович

/Ф.И.О. /

Руководитель: преподаватель, Цуроев Илез Магометович

/должность, Ф.И.О./

Дата защиты: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Магас – 2023

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1. ПОНЯТИЕ, ВИДЫ И МЕТОДЫ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

1.1 Понятие интерфейса пользователя и его типы 7

1.2 Методы и средства разработки пользовательского интерфейса 16

2. АНАЛИЗ ВИЗУАЛИЗАЦИЯ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

2.1 Анализ предметной области 28

2.2 Анализ определения и составления схем приложения 31

2.3 Количественная оценка и построение прототипа интерфейса 36

ЗАКЛЮЧЕНИЕ 41

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43

**ВВЕДЕНИЕ**

В наше время при разработке программного продукта очень важна скорость обработки информации, доступность, хранение и передача. Однако, не малую роль играет создание пользовательского интерфейса, так как именно он определяет, насколько легко и удобно будет пользоваться программой. Пользовательский интерфейс представляет собой совокупность используемых в программе средств ввода данных, способов отображения информации и элементов управления. Эффективность работы программного продукта определяется не только его функциональными возможностями, но и доступностью этих возможностей. Поэтому создание интуитивно-понятного дружественного интерфейса является важной задачей при разработке программного продукта.

В наше время при разработке программного продукта очень важна скорость обработки информации, доступность, хранение и передача. Однако, не малую роль играет создание пользовательского интерфейса, так как именно он определяет, насколько легко и удобно будет пользоваться программой. Пользовательский интерфейс представляет собой совокупность используемых в программе средств ввода данных, способов отображения информации и элементов управления. Эффективность работы программного продукта определяется не только его функциональными возможностями, но и доступностью этих возможностей. Поэтому создание интуитивно-понятного дружественного интерфейса является важной задачей при разработке программного продукта.

Сфера услуг в настоящее время является одной из важных отраслей народного хозяйства призванной удовлетворять индивидуальные запросы и потребности населения страны в различных видах услуг. Сфера услуг как отрасль экономической деятельности представляет собой совокупность организаций, цель которых - оказание разнообразных платных услуг по индивидуальным заказам населения. Таким образом, сфера услуг решает важнейшие социально-экономические задачи и ее значение в жизни общества неуклонно возрастает. Одним из видов таких услуг являются услуги автосалона.

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.

**Актуальность** данной темы состоит в том, что именно интерфейс видит пользователь и именно по интерфейсу, он оценивает весь продукт.

В наше время, время информационных технологий, все разработчики, архитекторы и дизайнеры стремятся к тому, чтобы их творения были не только полезными и эффективными, но и к тому, чтобы их использование было предельно удобным, понятным и комфортным.

**Целью** данной работы является обновление пользовательского интерфейса, а также наглядно представления о том, что такое пользовательский интерфейс. Данная информационная система должна существенно упростить работу сотрудника.

**Объектом** работы являются информационные технологии, а предметом - пользовательский интерфейс.

В ходе работы решаются следующие задачи:

* 1. Раскрытие понятия «пользовательского интерфейса», его
  2. Виды, типы, методы и средства разработки пользовательского интерфейса;
  3. Изучение и анализ пользовательского интерфейса;
  4. Внедрение обновленного интерфейса;

В процессе работы были использованы метод научного анализа и синтеза. Работа состоит из введения, двух глав, заключения, списка использованной литературы.

**Объектом** моего исследования являются сайты, а субъектом — их графические интерфейсы и целевые аудитории. Актуальность и новизна моей темы определяются тем, что ежедневно создается огромное количество сайтов, но создание качественного дизайна невозможно без учета приоритетов и интересов потенциальной целевой аудитории.

Основной целью работы является определение зависимости дизайна сайта от его целевой аудитории. Для достижения поставленной цели мне необходимо решить следующие задачи:

1. Выявить особенности сайтов, предназначенных для разных социальных групп.
2. Определить взаимосвязь целевой аудитории и дизайна интерфейса.
3. Методы исследования: теоретический анализ и обобщение ресурсов, описание, синтез данных.

**Практическая значимость** заключается в возможности применения результатов исследования при создании интерфейса.

**Предмет исследования** – пользовательский интерфейс и этапы его развития.

**Цель** выпускной квалификационной работы:

1. Обеспечение эффективного и удобного взаимодействия пользователей с программным продуктом.
2. Снижение нагрузки на пользователей при выполнении рутинных задач.
3. Использование простых и интуитивно понятных действий для выполнения сложных задач
4. Предоставление быстрого и понятного доступа к функциям программы.

**Задачи** выпускной квалификационной работы:

1. Создание удобного и эргономичного интерфейса программы.
2. Разработка и реализация функций, которые упрощают работу пользователя с программой.
3. Создание системы подсказок и инструкций для улучшения пользователя интерфейса.
4. Создание простой и удобной навигации по интерфейсу.
5. Улучшение производительности и быстродействия интерфейса программы.

**Структура Работы интерфейса**

Пользовательский интерфейс состоит из трёх основных частей:

1. Внешнее оформление, которое отвечает за представление информации оператору.
2. Функциональные возможности системы, включающие набор возможностей для эффективного выполнения его профессиональной и трудовой деятельности.
3. Техники взаимодействия оператора с программным интерфейсом.

**Актуальность** выпускной квалификационной работы: актуальность данной темы состоит в том, что именно в интерфейсе пользователя должны быть использованы современные программные обеспечения должны запускаться на различных платформах, у каждой из которых свои аппаратные особенности и возможности, многие компании разрабатывают продукт непосредственно под каждую платформу, тем самым увеличивая финансовые и временные затраты на разработку и поддержку в несколько раз.

**1. ПОНЯТИЕ, ВИДЫ И МЕТОДЫ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА**

***1.1 Понятие интерфейса пользователя и его типы***

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 [4; c.54].

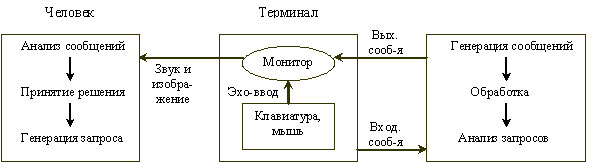


Рисунок 1. Взаимодействие пользователя с компьютером

В основном пользователь генерирует сообщения следующих типов:

1. Запрос информации;
2. Запрос помощи;
3. Запрос операции или функции;
4. Ввод или изменение информации;
5. В ответ пользователь получает подсказки или справки; информационные сообщения, требующие ответа; приказы, требующие действия; сообщения об ошибках и другую информацию.
6. Интерфейс пользователя компьютерного приложения включает:
7. Средства отображения информации, отображаемую информацию, форматы и коды;
8. Командные режимы, язык "пользователь - интерфейс";
9. Устройства и технологии ввода данных;
10. Диалоги, взаимодействие и транзакции между пользователем и компьютером, обратную связь с пользователем;
11. Поддержку принятия решений в конкретной предметной области;
12. Порядок использования программы и документацию на неё.

Пользовательский интерфейс 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.

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

1. Набор задач пользователя, которые он решает при помощи системы;
2. Используемая системой метафора (например, рабочий стол в MS Windows®);
3. Элементы управления системой;
4. Навигация между блоками системы;
5. Визуальный (и не только) дизайн экранов программы;
6. Средства отображения информации, отображаемая информация и форматы;
7. Устройства и технологии ввода данных;
8. Диалоги, взаимодействие и транзакции между пользователем и компьютером;
9. Обратная связь с пользователем;
10. Поддержка принятия решений в конкретной предметной области;
11. Порядок использования программы и документация на нее.

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.
2. WIMP - интерфейс (Windows - 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 "посредственно", 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, и поэтому его применяют в основном для военных целей.

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.

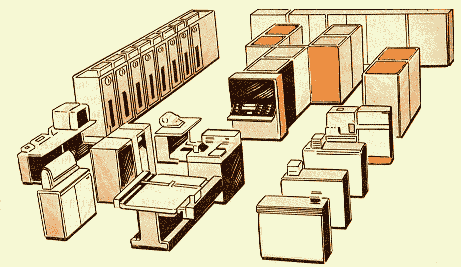


Рис.2. Вид большой ЭВМ серии ЕС ЭВМ

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 [7; c.88].

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 много платформенной операционной системой CP / M.

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.

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.

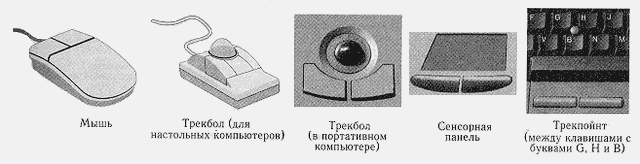


Рис.3. Манипуляторы

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 [9; c.13].

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.

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.

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:

1. "Проснись" - включение голосового интерфейса.
2. "Отдыхай" - выключение речевого интерфейса.
3. "Открыть" - переход в режим вызова той или иной программы. Имя программы называется в следующем слове.
4. "Буду диктовать" - переход из режима команд в режим набора текста голосом.
5. "Режим команд" - возврат в режим подачи команд голосом и некоторые другие.
6. 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.
7. "Речевая" технология является простейшей реализацией SILK - интерфейса.

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.

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) [14; c.54].

Интерфейсы пользователя бывают двух типов:

1. Процедурно-ориентированные:
2. Примитивные;
3. Меню;
4. Со свободной навигацией;
5. Объектно-ориентированные:
6. 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 одно документные (SDI) и многодокументные (MDI) интерфейсы.

Процедурно-ориентированные интерфейсы:

1. Обеспечивают пользователю функции, необходимые для выполнения задач;
2. Акцент делается на задачи;
3. Пиктограммы представляют приложения, окна или операции;
4. Содержание папок и справочников отражается с помощью таблицы-списка.

Объектно-ориентированные интерфейсы:

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.

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 [19].

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:

1. каждое окно меню занимает весь экран;
2. на экране одновременно присутствуют несколько разноуровневых меню (Windows).

В условиях ограниченной навигации, независимо от варианта реализации, поиск пункта более чем двухуровневого 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).

***1.2 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 [15; c.63].

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

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. Рынок программных средств предлагает широкий выбор инструментария для его разработки. Однако имеющиеся инструментальные средства поддерживают разработку только некоторых составляющих пользовательского интерфейса средствами четвертого поколения, остальные его составляющие программируются разработчиком, что значительно увеличивает затраты, сложность разработки и сопровождения.

Исследования, связанные с разработкой пользовательского интерфейса, начались с появления специальных операторов ввода-вывода в языках программирования и в настоящее время привели к появлению специализированных инструментальных средств для разработки интерфейса.

В литературе не существует единой общепринятой классификации средств для разработки пользовательского интерфейса. Так, в программное обеспечение для разработки пользовательского интерфейса разделяется 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 авторы относят 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.

В инструментарий для разработки интерфейса разделен на три группы, которые определяются следующим образом. В первую группу входит инструментарий для поддержки создания интерфейса написанием кода – UIMS и Toolkits; во вторую – интерактивные инструментальные средства, позволяющие сконструировать интерфейс из “заготовок” (кнопок, меню, полос прокрутки и т.д.), – Interface Builders; третий тип основан на создании интерфейса путем связывания отдельно созданных его компонент – Component Architectures [4; c.88].

Как замечено, терминология данного направления окончательно не сформировалась и в настоящее время также является предметом исследования. Однако в большинстве работ для ссылки на специализированные средства для разработки интерфейса приводится термин СУПИ, который и будет использоваться в данной работе.

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.

5. 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.

6. 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.

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

Поведение интерфейса и прикладной программы определяется характером взаимодействия с пользователем. Можно выделить три различных типа взаимодействия:

1. Инициатива диалога принадлежит пользователю, прикладной программе либо является смешанной.
2. Инициатива управления пользователем. Данный тип управления означает, что интерфейс предоставляет инициативу пользователю (прикладная программа так устроена) либо пользователь сам берет инициативу на себя, а интерфейс поддерживает такую возможность (прикладная программа так устроена).
3. Инициатива управления прикладной программой. Данный тип управления означает, что если прикладной программе необходима некоторая информация, то она запрашивает ее у пользователя, пользователь включается в процесс решения, когда необходимо ввести данные, требуемые системе.

Смешанная инициатива управления. Данный тип взаимодействия объединяет два предыдущих подхода, при котором пользователь определяет входные данные, но если прикладной программе для решения необходимы дополнительные данные, то она запрашивает их у пользователя.

Таким образом, в настоящее время существует большое количество инструментальных средств для разработки интерфейса, поддерживающих различные методы его реализации. Однако отсутствует единая общепринятая классификация предлагаемого инструментария, что затрудняет сравнение существующих средств между собой и выбор пользователями конкретного инструментального средства. Поэтому прежде, чем приступить к рассмотрению и сравнению инструментальных средств, следует ответить на следующие вопросы: имеются ли в инструментарии средства четвертого поколения для спецификации составляющих пользовательского интерфейса и как средствами четвертого поколения поддерживается разработка каждой составляющей пользовательского интерфейса?

Важность ответа на первый вопрос обусловлена актуальностью разработки инструментальных средств, позволяющих снизить стоимость разработки и сопровождения приложений, создаваемых с их помощью. Решением проблемы является использование языков четвертого поколения,

позволяющих разработчику специфицировать компоненты программного средства на высоком уровне, и затем по спецификации разработчика автоматически генерировать исполнимый код [10; c.32].

Для ответа на второй вопрос необходимо выделить составляющие пользовательского интерфейса, то есть те аспекты, по которым можно сравнивать интерфейсы между собой. При этом будем придерживаться следующих принципов:

1. Пользовательский интерфейс должен быть ориентирован на конечного пользователя и разрабатываться в соответствии с его требованиями;
2. Пользовательский интерфейс и прикладная программа, для которой он предназначен, разрабатываются раздельно.
3. Составляющие пользовательского интерфейса определяются принципами, указанными выше, а также выполняемыми им функциями.

По определению, например, в пользовательский интерфейс предназначен для обеспечения взаимодействия между пользователем и процессом, выполняющим некоторое задание – прикладной программой. Задачами данного взаимодействия является передача информации (исходных данных) от пользователя прикладной программе, выходных данных (результатов работы программы) пользователю. В соответствии с функцией интерфейса является также объяснение результатов работы прикладной программы, что до недавнего времени являлось характерной особенностью лишь интерфейсов экспертных систем [8; c.64].

Ориентация на конечного пользователя означает, что интерфейс должен иметь возможности для представления исходных данных и результатов в виде, общепринятом в данной предметной области, либо в зависимости от категорий пользователей и их пожеланий: графическом, табличном, вербальном, причем каждое из них также может иметь несколько видов представлений. Иными словами, как отмечено для одной и той же информации могут существовать различные передающие сообщения, образующие класс эквивалентных сообщений. При этом всегда существует базисная система сообщений, в которой можно выразить любую информацию о предметной области, однозначно понимаемую и интерпретируемую всеми ее представителями, и к которой сводятся все сообщения пользователя. Такой системой сообщений является система понятий предметной области. В терминах системы понятий именуются объекты предметной области, формулируются утверждения о том, что они

обладают некими свойствами и характеристиками, которые позволяют устанавливать сходство и различие объекта по отношению к другим объектам, а также указывают на взаимоотношения, в которых объекты находятся между собой. Таким образом, составляющей пользовательского интерфейса является описание информации через систему понятий предметной области, задающей функцию интерпретации сообщений.

Как было отмечено выше, информация для пользователя может представляться в виде сообщений (вербальных, графических, табличных), каждое из которых может принимать различные формы. Таким образом, в интерфейсе сообщения, передающие одну и ту же информацию для пользователя и прикладной программы, представляются по-разному: для пользователя сообщения формируются в виде, удобном ему либо принятом в его предметной области, для прикладной программы сообщениями являются значения переменных прикладной программы. Очевидно, что определение множества переменных прикладной программы сводится к определению имен, типов и способа представления их возможных значений [1; c.44].

Наряду с передачей сообщений пользователю в интерфейсе необходимо задание атрибутов, которые информацию не передают, но создают ему комфорт и удобство; их можно объединить общим термином дизайн интерфейса. К таким атрибутам относятся: расположение сообщений на экране, их размер, цвет и т.п., а также задание физических устройств ввода (клавиатура, манипуляторы, речевой ввод, машинное зрение и др.) и вывода (монитор, звук, фотографический вывод и др.). Таким образом, составной частью пользовательского интерфейса, неразрывно связанной с передачей сообщений, является определение формы сообщений [3; c.55].

Интерфейс должен выполнять преобразование введенной пользователем информации, представленной в виде понятных ему сообщений в значения переменных прикладной программы, а также значений переменных прикладной программы, которые являются результатами ее работы к сообщениям пользователю. Для преобразования информации пользователю в различные сообщения в составе интерфейса необходим блок интеллектуальной поддержки пользователя, который контролирует возможные ошибки, формирует объяснения, управляет системой помощи [6; c.88].

Любое взаимодействие двух или нескольких объектов между собой (в данном случае пользователя и интерфейса) всегда подчиняется определенным правилам. Правила взаимодействия пользователя и интерфейса также необходимо определять в интерфейсе. Эти правила должны задавать последовательность переходов от одного состояния к

другому. Соответственно, взаимодействие интерфейса с пользователем должно содержать правила обмена сообщениями (в данном случае это действия пользователя и интерфейса по управлению исходными данными и результатами).

Таким образом, в состав пользовательского интерфейса входят:

1. Базисная система сообщений (система понятий предметной области);
2. Система сообщений для пользователя;  
   система сообщений для прикладной программы;  
   средства обеспечения удобства и комфорта работы пользователя;
3. Средства интеллектуальной поддержки пользователя;  
   средства управления взаимодействием пользователя и интерфейса.

Рассмотрим, как поддерживается разработка каждой составляющей пользовательского интерфейса средствами четвертого поколения.

Поддержка описания системы понятий предложена в. По спецификации системы понятий, для которой предлагается специализированный язык, автоматически генерируются сообщения, представляемые в вербальном виде множеством каскадных меню и окон. Недостаток данной спецификации заключается в том, что структура системы понятий предметной области ограничена иерархическим представлением, а ее описание выполняется на специализированном языке в пакетном режиме [12; c.88].

В работах также предложено начинать проектирование интерфейса с моделирования задачи и предметной области. Для этого пользователю предлагается на неформальном языке описать постановку задачи, из которой автоматически выделяются понятия предметной области и действия с ними. Следующими этапами является формализация полученной постановки задачи путем отсеивания ненужных элементов, организация классов выделенных элементов, задание области и типов их допустимых значений, действий над ними с целью создания полноценной модели предметной области. В качестве преимуществ подобного способа извлечения задачи авторы указывают на снижение степени непонимания между разработчиком и пользователем, вовлечение пользователя в проект с самого начала его реализации и построение им каркаса модели задачи и модели предметной области. Однако вызывает сомнение возможность использования данного подхода для решения задач со сложной моделью предметной области, имеющей большой

объем и сложную структуру системы понятий, необходимую для решения задачи, обеспечения пользователя интеллектуальной поддержкой, поскольку каркас и элементы модели (термины и понятия) выделяются на основе неформального описания задачи пользователем. Наш опыт проектирования сложных систем, например экспертной системы “Консультант-2″, и в частности, ее интерфейса, показал, что процесс формирования системы понятий, бесспорно, должен осуществляться при активном участии высококвалифицированных специалистов предметной области на основе серьезного предшествующего ее анализа с целью последующей формализации. Инструментарий для проектирования интерфейса поэтому должен быть ориентирован скорее на разработчика интерфейса, чем на конечного его пользователя [13; c.77].

Инструментальные средства типа Toolkits предлагают библиотеки интерфейсных элементов, используемых в диалоге, таких как панели диалога, формы, различные типы меню, представление иерархии данных в виде ветвящейся структуры и т.п. При этом разработчик имеет не только возможность выбора необходимых интерфейсных элементов, но также и возможность организации сложных комплексов из предлагаемых базовых примитивов средствами визуального и объектно-ориентированного программирования. Однако трудно говорить о поддержке конструирования интерфейсов, поскольку предлагаемые библиотеки отражают довольно произвольное мнение о стандартах элементов интерфейса, не используя специфику приложений, для которых применение библиотек оправдано. [16; c.105]

Следует отметить, что во всех существующих Toolkits отсутствуют специальные средства для проектирования пользовательского интерфейса исходя из его составляющих. Поэтому разработчики интерфейса вынуждены проектировать все его части вместе, явно не отделяя одну составляющую от другой, хотя проектирование различных его составляющих требует использования различных типов понятий и уровней абстракции. Технология разработки интерфейса данными средствами организована таким образом, что разработчик выбирает интерфейсный элемент и “нанизывает” на него содержание интерфейса, а не наоборот, в соответствии со структурой и содержанием (системой понятий) предлагаются формы ее представления (возможно, автоматически формируются). Разрабатывая таким образом интерфейс, его разработчик должен корректировать структуру и содержание исходных данных под формы, предлагаемые в инструментальном средстве. [4; c.76].

Для исходных данных, представленных графически, имеется множество графических пакетов векторной и растровой графики. Графические пакеты позволяют только формировать изображения, но не имеют средств для связи графических и вербальных описаний (связи системы понятий и системы сообщений), поэтому данную часть интерфейса приходится программировать. Следует напомнить, что речь идет о средствах четвертого поколения, которые позволяют на высоком уровне специфицировать интерфейс.

Сделана попытка связать систему понятий и систему сообщений. Для этого в инструментарии имеется база данных, в которой хранится информация о том, какими интерфейсными элементами удобнее представлять те или иные виды данных. На основе этой базы данных разработчик может назначать примитивы элементам либо группам данных и затем автоматически генерировать прототип интерфейса. Такой подход удобен для вербального представления данных, однако графическое представление зависит от предметной области, поэтому в данном случае предлагаемые в базе данных примитивы не могут быть использованы для формирования сообщений.

Все классы инструментальных средств поддерживают разнообразные возможности задания параметров комфортности интерфейса средствами визуального и объектно-ориентированного программирования, позволяющие задать расположение интерфейсных элементов на экране монитора, их цвет, текстуру, размер и др. в зависимости от требований пользователей, психологии и эргономики, а также определить физические устройства ввода/вывода информации.

Для организации взаимодействия пользователя и интерфейса в настоящее время не известно специальных возможностей, которые бы позволяли разработчику на уровне спецификации определить действия пользователя по управлению исходными данными, поэтому разработчику приходится программировать данную составляющую интерфейса. В работе инструментарий предлагает разработчику возможность сохранять наборы исходных данных, их просмотр, редактирование для последующего ввода.

Множество переменных и представление их значений обычно приходится программировать, как-либо они формируются по жестко заданным в инструментарии правилам [17].

Средства генерации объяснений результатов работы программной системы представлены в работе. Для этого разработчикам интерфейса предлагается специальный макроязык, на котором они могут описать шаблон объяснения. Однако данный язык позволяет представить объяснение только в вербальном виде, имеет ограниченные средства по форматированию текста объяснения и содержит ограничение на формат результатов работы программной системы – только в виде кортежей отношений. Множество других систем генерации объяснений ориентированы исключительно на экспертные системы, они зависят от машины логического вывода, а также требуют включения дополнительных знаний в базу знаний. В авторы предлагают средства для автоматической генерации средств помощи по представлению базы знаний [18].

Взаимодействие интерфейса и прикладной программы не поддерживается на высоком уровне, а программируется разработчиком.

Итак, основной целью СУПИ является снижение затрат на создание и сопровождение пользовательского интерфейса, которое достигается предоставлением средств высокого уровня для определения интерфейса и освобождением таким образом разработчика от низкоуровневого программирования.

**2. АНАЛИЗ ВИЗУАЛИЗАЦИЯ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ**

***2.1 Анализ предметной области***

Сфера услуг в настоящее время является одной из важных отраслей народного хозяйства призванной удовлетворять индивидуальные запросы и потребности населения страны в различных видах услуг. Сфера услуг как отрасль экономической деятельности представляет собой совокупность организаций, цель которых – оказание разнообразных платных услуг по индивидуальным заказам населения. Таким образом, сфера услуг решает важнейшие социально-экономические задачи, и ее значение в жизни общества неуклонно возрастает. Одним из видов таких услуг являются услуги автосервиса.

Предлагаемая программа реализует информационную систему, предназначенную для организации учета распределения работ, а также ведет учет стоимости произведенных работ. Приложение позволяет производить ввод, редактирование и просмотр содержимого баз данных, а также отвечать на запросы пользователя.

Данная информационная система может применяться в автосервисах. Она легка в обращении, позволяет хранить сведения в одной базе данных, экономит рабочее время за счет автоматизации некоторых процессов таких, как учет стоимости работ. Поэтому предлагаемая программа должна существенно упростить работу автосервисов.

При анализе информации формируют профили пользователей. Профилями называют описание главных категорий пользователя. Потенциальными пользователями в этой работе являются консультант-механик и менеджер по направлению товара.

Сегодня ни для кого не секрет, что графический пользовательский интерфейс любого программного продукта является одним из ключевых факторов его популярности. Времена, когда связь между пользователем и приложением повсеместно устанавливалась при помощи командной строки, практически безвозвратно канули в лету, уступив место графическому пользовательскому интерфейсу (GUI от Graphical User Interface). Создание грамотного пользовательского интерфейса - процесс трудоемкий и требующий максимального внимания к деталям. Создаваемый интерфейс должен максимально реализовывать возможности программы, но вместе с тем не перегружать пользователя обилием меню, кнопок, изображений и текста. Даже самое мощное программное обеспечение, спроектированное талантливейшими инженерами и написанное самыми искусными программистами, без удобной организации взаимодействия с пользователем рискует так и остаться невостребованным. Краеугольным камнем удобства программы либо интернет-сайта является его быстрота, которая достигается путем подробного изучения функций продукта и разработки максимально эффективного, интуитивно-понятного меню навигации, грамотным использованием иконок (ico), при котором пользователь не тратит время на поиски требуемой ссылки.

На практике программирование Windows-приложений предполагает экстенсивное использование различных инструментальных средств и мастеров, которые намного упрощают этот процесс однако все указанные средства автоматизации заслоняют то, что лежит в основе создания графического пользовательского интерфейса поэтому сначала мы рассмотрим основы создания графических пользовательских интерфейсов Иными словами, мы научимся создавать простые приложения Windows с самого начала, пользуясь только комплексом инструментальных средств разработки программ NET Framework SDK Это значит, что вначале мы будем создавать простые приложения Windows без применения каких-либо специальных сервисных программ Будут рассмотрены основы рисования с помощью Windows Forms (Формы Windows) с применением шрифтов и кистей, а также необходимые обработчики событий Мы объясним принципы обработки событий в Windows Forms (Формы Windows) и реализуем обработчики событий мыши.

С помощью Windows Forms (Формы Windows) мы также реализуем меню и соответствующие обработчики событий. Кроме того, мы рассмотрим управляющие элементы, а после этого изучим среду Visual Studio.net, посредством которой можно без труда создать простой графический пользовательский интерфейс на С#.

В нашем случае обязательным заданием являлась реализация трех кнопок (методов):

1. добавление;
2. изменение/обновление;
3. удаление.

Таблица 1.1. Профили пользователей

|  |  |  |
| --- | --- | --- |
| Пользователь | Консультант-механик | Менеджер по направлению  товара |
| Социальные  характеристики | Русскоязычные  Низкий уровень владения  компьютером  Знание техники | Русскоязычные  Средний уровень владения  компьютером  Наличие знаний в области техники, торговли, логистики |
| Мотивационная среда | Производственная необходимость  Низкая мотивация к обучению | Производственная необходимость  Высокая мотивация к обучению |
| Навыки и умения | Прошли предварительный тренинг с программой | Имеют навыки работы с программой |
| Требования к ПО | Возможность использования в локальной сети  Обеспечение информацией о наличии автозапчастей и об оказываемых услугах  Возможность просмотра и изменения информации о состоянии автомобилей  Возможность формирования заказов | Возможность использования в сети Интернет  Возможность просмотра и изменения информации о наличии автозапчастей или их необходимости |
| Задачи пользователя | Просмотр, сортировка, поиск информации по автомобилям и  автозапчастям  Изменение статуса выполнения заказа  Сохранение отчета по выполненному заказу | Просмотр, сортировка, поиск, изменение информации по автозапчастям |
| Рабочая среда | Стандартизированные ПК, локальная сеть | Стандартизированные ПК, локальная сеть, Интернет |

В ней разработаны основные критерии, по которым оценивают и выбирают фотокамеры:

1. цена;
2. производитель;
3. тип;
4. габариты;
5. вес;
6. светочувствительность диафрагмы;
7. скорость затвора/выдержка и другие.

***2.2 Анализ определения и составления схем приложения***

Проделав анализ предметной области, можно выделить функциональность приложения, исходя из целей и задач пользователя. Рассмотрим определение функциональности на примере одного из профилей: оператор. Исходя из задач этой категории пользователей, можно сформировать следующий перечень функций необходимых в приложении:

1. Создание заказа;
2. Поиск заказа;
3. Редактирование заказа;
4. Просмотр списка автозапчастей;
5. Поиск услуг и деталей;
6. Поиск автомобиля;
7. Просмотр данных по заказу;
8. Просмотр данных по автозапчасти;
9. Добавление данных по услугам/автозапчастям в заказ;
10. Сохранение состояния выполнения заказа;
11. Сохранение заказа.

Для выделения профиля пользователей составлены пользовательские сценарии, чтобы рассмотреть варианты действий, выполняемые им.

Механик создает новый заказ. Вводит данные клиента. Затем, используя фильтры, ищет и добавляет в заказ необходимые детали и услуги. Сохраняет заказ.

Механик выполняет поиск заказа по данным клиента и/или автомобиля. Затем редактирует поле состояния выполнения заказа и сохраняет изменения.

Механик выполняет поиск заказа по нужным атрибутам. Затем просматривает заказ и приступает к его выполнению.

Проанализировав предметную область и выделив профили потенциальных пользователей, можно определить операции пользователей, которые он может выполнять в рамках предложенной им приложением:

1. Создать новый заказ;
2. Отсортировать заказы;
3. Найти заказ по текущим атрибутам поиска;
4. Задать атрибуты поиска услуг;
5. Выбрать заказ из списка;
6. Открыть заказ для редактирования;
7. Удалить заказ;
8. Редактировать данные клиента;
9. Открыть список деталей;
10. Открыть список услуг;
11. Сохранить заказ;
12. Возврат в журнал заказов без сохранения;
13. Отметить выполненные услуги;
14. Отфильтровать услуги;
15. Просмотреть список услуг;
16. Выбрать услугу из списка;
17. Добавить услугу в заказ;

Просмотреть список деталей;

1. Отфильтровать детали;
2. Выбрать деталь из списка;
3. Добавить услугу установки;
4. Добавить деталь в заказ;
5. возврат в окно текущего заказа.

Проделав анализ предметной области, сформировав профили и сценарии действий пользователей, определив функциональность приложения можно составить схему навигационной системы.

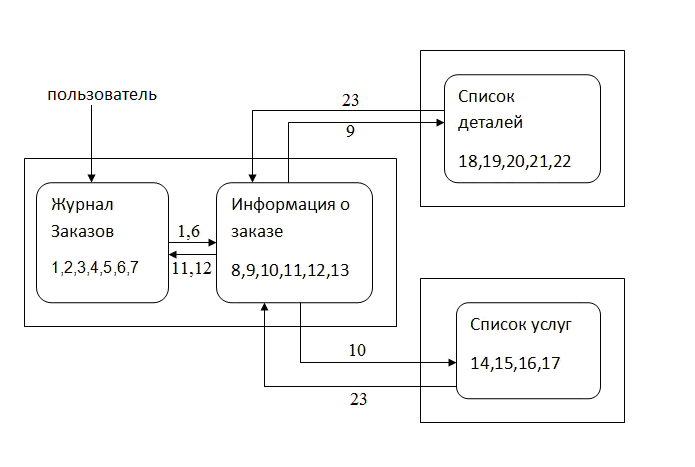


Рис. 1.1. Схема навигации

В данной схеме три функциональных блока и четыре основные экранные формы:

1. Журнал заказов;
2. Текущий заказ;
3. Список автомобилей и деталей;
4. Список услуг.

Данные функциональные блоки соответствуют работе пользователей с информацией:

1. По заказам;
2. По деталям;
3. По услугам.

Цифрами на рисунке указаны отдельные операции, выполняемые пользователями.

Проектирование пользовательских диалогов

Сценарий 1.

1. Ввод команды на создание нового заказа;
2. Ввод данных о клиенте;
3. Ввод команды на открытие списка деталей;
4. Задание параметров фильтров;
5. Просмотр списка;
6. Выбор детали из списка;
7. Ввод команды на добавление услуги установки;
8. Ввод команды на добавление детали в заказ;
9. Ввод команды на возврат в окно текущего заказа;
10. Ввод команды на открытие списка услуг;
11. Задание параметров фильтров;
12. Просмотр списка услуг;
13. Выбор услуги;
14. Ввод команды на добавление услуги;
15. Ввод команды на возврат в окно текущего заказа;
16. Ввод команды на сохранение заказа.

Сценарий 2.

1. Задание атрибутов поиска заказа;
2. Выбор заказа из списка;
3. Ввод команды на открытие заказа;
4. Отметка о выполнении услуги;
5. Ввод команды на сохранение заказа.

Сценарий 3.

1. Задание атрибутов поиска услуг;
2. Выбор заказа из списка;
3. Ввод команды на открытие заказа;
4. Ввод команды на возврат в журнал заказов.

Начало ЖЗ

Начало ЖЗ

3 ЖЗ

5 ЖЗ

6 ЖЗ-ТЗ

13 ТЗ

11 ТЗ-ЖЗ

4 ЖЗ

5 ЖЗ

6 ЖЗ-ТЗ

12 ЖЗ-ТЗ

Конец ЖЗ

Конец ЖЗ

Начало ЖЗ

1 ЖЗ-ТЗ

8 ТЗ

9 ТЗ-СД

19 СД

18 СД

21 СД

20 СД

22 СД

23 СД-ТЗ

10 ТЗ-СУ

14 СУ

15 СУ

16 СУ

17 СУ

23 СУ-ТЗ

22 ТЗ-ЖЗ

Конец ЖЗ

а)

б)

в)

Рисунок 1.1

а) Для сценария 1

б) Для сценария 2

в) Для сценария 3

Рисунок 1.1. Схемы диалогов

На этапе предварительного и высокоуровневого проектирования пользовательского интерфейса был проведен анализ предметной области, были определены профили потенциальных пользователей, определена функциональность приложения [5; c.66].

Далее был определен основной профиль- механик, для которого составлены сценарии и операции пользователя. По каждому сценарию были составлены схемы действия, при помощи которых определили необходимость каждой функции для работы приложения.

***2.3 Количественная оценка и построение прототипа интерфейса***

Пройдя этап предварительного и высокоуровневого проектирования интерфейса пользователя, то есть, проанализировав предметную область, сформировав профили и сценарии действий пользователей, составив схему навигаций и определив операции, можно разработать модели прототипов экранных форм для каждого из функциональных блоков.

Исходя из того, что прототип — это возможность предоставить функционирующий продукт на обозрение, я могу подобрать такие кейсы использования:

1. демонстрация клиенту;
2. демонстрация пользователям для сбора качественных и количественных данных;
3. демонстрация коллегам для получения обратной связи;
4. демонстрация разработчикам для объяснения логики любого пользовательского пути в интерфейсе.

Прототип помогает не тратить время на разработку непроверенного функционала. Вы показываете прототип пользователям, они дают вам обратную связь за короткое время. А разработчики не отвлекаются на лишнюю работу.

На начальном этапе проводится анализ проекта, составляется структура страниц, продумывается взаимодействие с посетителями сайта. В случае редизайна дополнительно выявляются недочеты в удобстве использования.

При создании прототипа мы придерживаемся этих принципов:

1. **Цели посетителей**. Они позволяют не забывать, для решение какой задачи человек приходит на сайт.
2. **Функциональность важнее эстетики**. Графическое оформление отвлекает от сути, прототип должен помочь определится с тем, как будет решаться задача, а дизайн будет проработан позднее.
3. **Понятные разделы**. Разделов не должно быть очень много, и они должны называться так, как привык пользователь.
4. **Простота и доступность**. Понятность прототипа всем участникам процесса – дизайнеру, программисту, заказчику и будущему посетителю.

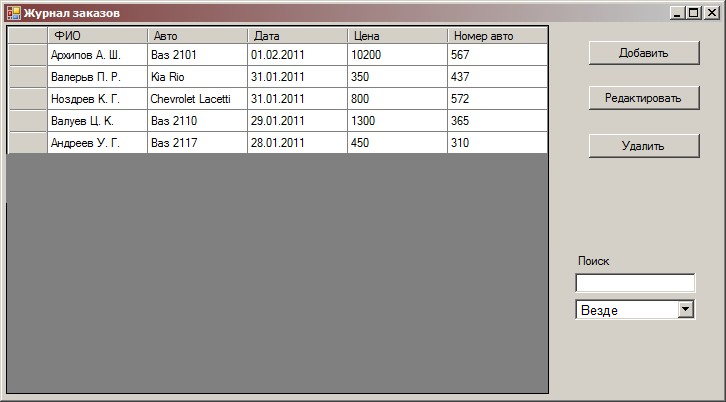


Рис. 2.1. Журнал заказов

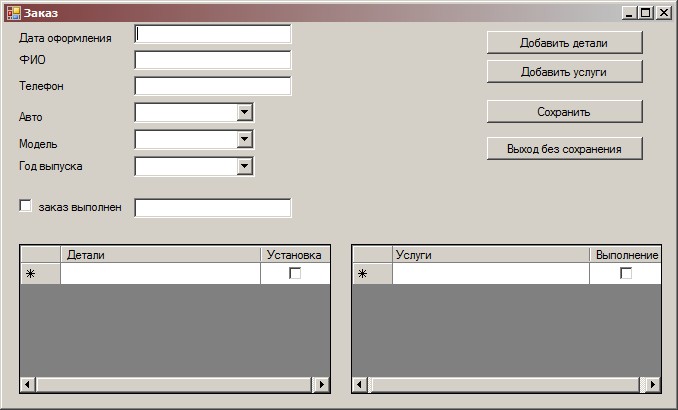


Рис. 2.2. Текущий заказ

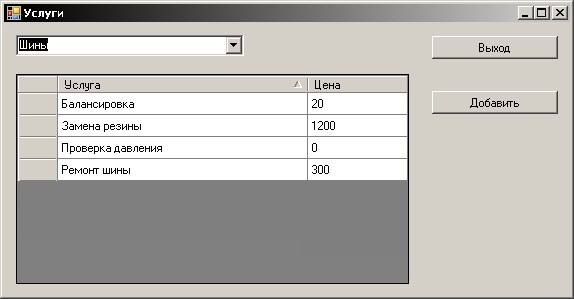


Рис. 2.3. Список услуг

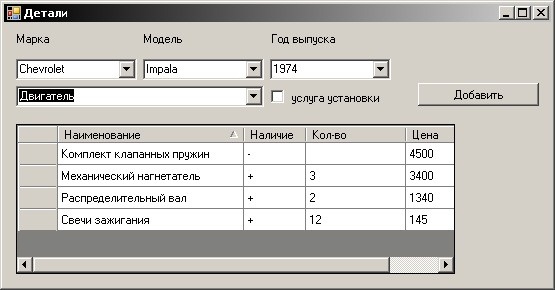


Рис. 2.4. Список деталей

Разработав электронные прототипы, можно провести количественную оценку элементов интерфейса. 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.

На этапе низкоуровневого проектирования пользовательского интерфейса была проведена количественная оценка интерфейса, которая показала, насколько быстро можно выполнить любую четко сформулированную задачу. С помощью правила GOMS было определено время всей работы.

Тестирование интерфейса является исключительно важной задачей при проектировании интерфейса. Начальный этап тестирования связан с разработкой прототипа интерфейса. На этом этапе используется имеющиеся результаты проектирования: общая схема приложения, планы отдельных экранных форм. Выяснив последовательность действий, описанной в сценарии, рассмотрев общую схему приложения, можно сформировать электронный прототип, который представляет собой демонстрационный ролик, выполненный в среде MS Power Point [20].

Тестирование проводится на представителях пользовательской

аудитории ранее не знакомых с разрабатываемым продуктом.

Тестовое задание: создать новый заказ. Ввести данные клиента: Смирнов В. Г., тел: 2737193, автомобиль: Chevrolet Impala 1974. Затем, используя фильтры, найти и добавить в заказ необходимые детали (свечи зажигания) и услуги (установка свечей, балансировка) и сохранить заказ.

1. Скорость выполнения задания
2. Первый тестируемый выполнил задание за 62 секунды.
3. Второй тестируемый выполнил задание за 49 секунд.
4. Расчетное время 47,2 секунды.
5. Комментарии пользователя

Тестируемый пользователь Булат:

1. «Интерфейс прост и понятен в обращении. Кнопка "выход без сохранения" непонятна, так как есть кнопка сохранить (т.е. если её не нажимать и просто нажать выход, то будет произведено то же самое действие). Нельзя выбрать количество, к примеру, свечей зажигания.»

Тестируемый пользователь Любовь:

1. «Обычная программа. Из минусов - непонятные пункты списка, такие как «Установка» и «Выполнение».
2. После выполнения задания, тестируемые пользователи воспроизвели по памяти электронные прототипы, которые представлены в Приложении Б.
3. Несмотря на то, что тестирование прошло легко, время, потребовавшееся пользователям на выполнение тестовых заданий, превышает расчетное.

Это связано с тем, что тестируемые не проходили предварительных тренингов по работе с программой.

**ЗАКЛЮЧЕНИЕ**

В ходе данной работы был спроектирован пользовательский интерфейс программы, предназначенной для работников автосервиса.

Были получены навыки проведения этапа низкоуровневого и высокоуровневого проектирования интерфейса, была проанализирована предметная область, выделены профили потенциальных пользователей, составлены схемы навигации и пользовательские сценарии. Также был получен опыт тестирования интерфейса.

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:

1. Интерфейс пользователя - набор методов взаимодействия компьютерной программы и пользователя этой программы.
2. Программный интерфейс - набор методов для взаимодействия между программами.

Тестирование интерфейса является исключительно важной задачей при проектировании интерфейса. Начальный этап тестирования связан с разработкой прототипа интерфейса. На этом этапе используется имеющиеся результаты проектирования: общая схема приложения, планы отдельных экранных форм. Выяснив последовательность действий, описанной в сценарии, рассмотрев общую схему приложения, можно сформировать электронный прототип, который представляет собой демонстрационный ролик, выполненный в среде MS Power Point.

Взаимодействие интерфейса и прикладной программы не поддерживается на высоком уровне, а программируется разработчиком.

Итак, основной целью СУПИ является снижение затрат на создание и сопровождение пользовательского интерфейса, которое достигается предоставлением средств высокого уровня для определения интерфейса и освобождением таким образом разработчика от низкоуровневого программирования.

На этапе предварительного и высокоуровневого проектирования пользовательского интерфейса был проведен анализ предметной области,

были определены профили потенциальных пользователей, определена функциональность приложения.

Далее был определен основной профиль-механик, для которого составлены сценарии и операции пользователя. По каждому сценарию были составлены схемы действия, при помощи которых определили необходимость каждой функции для работы приложения.

Пройдя этап предварительного и высокоуровневого проектирования интерфейса пользователя, то есть, проанализировав предметную область, сформировав профили и сценарии действий пользователей, составив схему навигаций и определив операции, можно разработать модели прототипов экранных форм для каждого из функциональных блоков.

**Анализ** пользовательского интерфейса - независимый метод анализа существующих систем в целях выявления пользовательских функциональных требований. Лучше всего взаимодействовать с существующими системами непосредственно, но при необходимости можно использовать снимки экрана. Руководства пользователя коробочных продуктов часто содержат снимки экрана, которые можно использовать в качестве отправного пункта. Если существующей системы нет, можно посмотреть на пользовательский интерфейс похожих продуктов.

**Актуальность** выпускной квалификационной работы: актуальность данной темы состоит в том, что именно в интерфейсе пользователя должны быть использованы современные программные обеспечения должны запускаться на различных платформах, у каждой из которых свои аппаратные особенности и возможности, многие компании разрабатывают продукт непосредственно под каждую платформу, тем самым увеличивая финансовые и временные затраты на разработку и поддержку в несколько раз.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

***Книги и сборники:***

1. Основы Web-дизайна. Самоучитель / В.В. Дунаев. - М.: БХВ-Петербург, **2019**. – 44 c.
2. Дэвид Макфарланд. Новая большая книга CSS. — М.: Питер, 2018.
3. Басов, К. А. Графический интерфейс комплекса ANSYS / К.А. Басов. - М.: ДМК Пресс, 2018. - 384 c.
4. Басов, К. Графический интерфейс комплекса ANSYS / К. Басов. - М.: Книга по Требованию, 2020. - 54 c.
5. Белоус, А. И. Биполярные микросхемы для интерфейсов систем автоматического управления / А.И. Белоус, О.Е. Блинков, А.А. Силин. - Москва: Наука, 2018. - 66 c.
6. Болл, С.Р. Аналоговые интерфейсы микроконтроллеров / С.Р. Болл. - М.: Додэка-XXI, 2021. - 88 c.
7. Ганеев, Р. М. Web-интерфейс баз данных ODBC / Р.М. Ганеев. - М.: Горячая линия - Телеком, 2020. - 88 c.
8. Головач, В.В. Дизайн пользовательского интерфейса (v 1.2) / В.В. Головач. - М.: [не указано], 2020. - 183 c.
9. Климов, А.П. MS Agent. Графические персонажи для интерфейсов (+ CD-ROM) / А.П. Климов. - М.: БХВ-Петербург, 2018. - 13 c.
10. Коутс, Р. Интерфейс "человек-компьютер" / Р. Коутс, И. Влейминк. - М.: Мир, 2021. - 32 c.
11. Введение в Web-дизайн. Учебное пособие/Алексеев А. — М.: ДМК Пресс, 2019.
12. Веб-дизайн. Элементы опыта взаимодействия/Гарретт Джесс. — М.: Символ-Плюс, 2020. – 88 c.
13. Веб-дизайн / С.В. Киселев. — М.: Academia, 2019. - 77c.
14. Куртов, Михаил Генезис графического пользовательского интерфейса. К теологии кода / Михаил Куртов. - М.: ТрансЛит, 2020. - 54 c.
15. Программирование Web-страниц / С.В. Глушаков, И.А. Жакин, Т.С. Хачиров. - М.: Феникс, 2018. – 63 c.
16. UI Design Principles [eBook] / Michael Filipiuk, 2018. – 105 с

***Электронные издания (интернет-ресурсы):***

1. Статьи по изучению HTML 5 и CSS 3 – <https://stackoverflow.com>.
2. Аналитические статьи по дизайну сайта – <https://m.habr.com> .
3. Курсы по дизайну веб-сайтов – <https://skillbox.ru>.
4. Статья по веб-дизайну – <https://ru.m.wikipedia.org>.
5. Изучение веб-дизайна – <https://github.com>.